Programare web

Laboratorul 00

0. Introducere

Cuprins

0. Introducere 1
0.1. Prezentare generală a lucrărilor de laborator 1
0.2. Începutul laboratorului 2
0.3. Finalul laboratorului 2

0.1. Prezentare generală a lucrărilor de laborator

Programul orelor de laborator, organizate pe săptămâni, sunt după cum urmează:

1. Sistemul de versionare Git (utilizare GitHub Classroom). Fișiere XML și validarea lor online
2. Pagină web HTML ca în imagine
3. Site web cu elemente semantice
4. Site web care să folosească stiluri CSS
5. Pagină web controlată prin JavaScript
6. Server HTTP care să permită vizualizarea conținutului directoarelor de pe harddisk
7. Site web de tipul Single Page Application prin intermediul AJAX
8. Test / Prezentare proiect 1
9. Prototipuri/clase, Promises, Web Storage, Web Worker, IndexedDB
10. Website cu nodeJS care să permită testarea unui utilizator cu ajutorul unui test grilă
11. Adăugarea logicii de autentificare a utilizatorilor (cookie, sesiuni)
12. Conectarea la baze de date
13. Securizarea aplicației / Prezentare proiect 2

Veți crea două proiecte pe GitHub care vor fi formate din temele din laboratoare: Proiect-1 (S1-S7) și Proiect-2 (S9-S13). Primul proiect are o tematică la alegere, iar al doilea va avea o tematică impusă.

Nota primită la sfârșitul laboratoarelor constă din media notelor primite pe cele două proiecte la care se pot adăuga sau deduce puncte în funcție de activitatea din cadrul laboratoarelor.

În evaluarea proiectelor se vor considera următoarele aspecte:

- Progresul și activitatea săptămânală de pe GitHub,
- Integrarea celor trei teme din fiecare laborator în tematica proiectului,
- Gradul de înțelegere a conceptelor predate și utilizate,
- Complexitatea proiectului,
- Aspectul vizual și ușurința în utilizare a site-ului.

Atenție! Pentru nota maximă 10 va trebui ca site-ul pe care îl veți realiza în cadrul proiectelor să aibă o anumită tematică, iar temele din cadrul fiecărui laborator trebuie adaptate temei de site alese.

Altfel, nota maximă este 8 dacă doar se respectă cerințele din laborator.

Pentru a obține nota 10 trebuie să fie implementate diferite funcționalități și utilizate concepte în plus față de ce este discutat în cadrul laboratorului.

0.2. Începutul laboratorului

Dacă nu mai știți adresa URL a proiectului de pe Git, atunci accesați adresa https://github.com/HDR-2023-2024 și, dacă sunteți autentificați, ar trebui să vă apară proiectul.

Dacă utilizați calculatoarele din laborator trebuie ca, la începutul fiecărui laborator (exceptând primul), să luați repository-ul vostru de pe git urmând pașii:

1. Asigurați-vă că nu sunt proiecte în directorul `/home/student`:
    a. Deschideți terminalul Linux (apăsați tasta windows ➜ scrieți terminal ➜ apăsați tasta Enter).
    b. Ar trebui să fiți în directorul ~, adică `/home/student`.
    c. Scrieți în terminal comanda: `rm -R -f numeProiect-*`
    De exemplu, `rm -R -f proiect-1-*`
2. Clonați proiectul de pe GitHub:
    a. În directorul ~ lansați comanda:
    `git clone https://github.com/HDR-2023-2024/numeProiect-numeUtilizatorGitHub.git`
    b. Se va crea un director cu numele *numeProiect-numeUtilizatorGitHub* în directorul ~.
3. Importați proiectul în Visual Studio Code
    a. Din meniul de sus din Visual Studio Code: File ➜ Open folder ➜ selectați directorul `/home/student/numeProiect-numeUtilizatorGitHub`.
    b. Pentru a vedea fișierele din directorul selectat se accesează fereastra Explorer prin apăsarea butonului de sub meniul File (sau, din meniul de sus din Visual Studio Code: View ➜ Explorer).
    c. Deschideți terminalul (CTRL + `) din Visual Studio Code și asigurați-vă că sunteți în directorul `/home/student/numeProiect-numeUtilizatorGitHub`.
4. Cel puțin după rezolvarea fiecărui task din laborator trebuie să adăugați modificările pe GitHub:
    a. Setați numele și adresa de email a persoanei care va face commit-urile pe acest repository (doar o singură dată înainte de primul commit!)
    `git config user.name "Prenume Nume"`
    `git config user.email "numeUtilizatorTUIASI@student.tuiasi.ro"`
    b. Adăugați starea CURENTĂ a fișierului specificat la index (aceste modificări vor fi procesate la următorul commit):
    `git add numeFisier`
    c. Creați un nou commit care conține toate modificările din index (tot ce a fost adăugat cu git add):
    `git commit -m "Rezolvare task ..."`
    d. Trimiteți toate modificările din repository-ul local la repository-ul de la GitHub:
    `git push -u origin main`

0.3. Finalul laboratorului

Pașii care trebuie urmați la sfârșitul fiecărui laborator (ultimul pas referitor la ștergerea proiectului de pe harddisk-ul local nu este necesar dacă lucrați pe propriul calculator):

1. Asigurați-vă că ați salvat pe GitHub proiectul:
    a. Verificați în Visual Studio Code să fie salvate toate fișierele (fișierele care nu sunt salvate au o bulină albă plină în dreapta numelui fișierului din tab-ul de sus - dacă fișierul este deschis).
    b. Verificați că au fost adăugate toate fișierele sub controlul versiunilor.
    c. Faceți commit (opțiunea -a adaugă sub controlul versiunilor toate fișierele care au fost adăugate anterior, dar între timp au fost modificate sau șterse):

`git commit -a -m "`*`mesaj commit`*`"`

d. Verificați dacă nu au rămas fișiere înafara controlului versiunilor:
`git status`
e. Încărcați pe GitHub proiectul:
`git push -u origin main`
f. Accesați din browser adresa `https://github.com/HDR-2023-2024/`*`numeProiect-numeUtilizatorGitHub`*`.git` și verificați că au fost încărcate toate modificările.

2. Ștergeți proiectul de pe calculatorul la care ați lucrat în cadrul laboratorului:
   a. Închideți Visual Studio Code.
   b. Deschideți terminalul Linux (apăsați tasta windows ➜ scrieți terminal ➜ apăsați tasta Enter).
   c. Scrieți în terminal comanda:
   `rm -R -f `*`numeProiect-numeUtilizatorGitHub`*

Programare web

Laboratorul 01

1. Sisteme de versionare. Limbajul XML

Cuprins

1. Sisteme de versionare. Limbajul XML 1
1.1. Concepte și tehnologii 1
1.2. Crearea proiectului inițial 1
1.2.1. Crearea contului pe GitHub cu adresa instituțională 1
1.2.2. Crearea proiectului pe site-ul GitHub 2
1.2.3. Utilizarea Git din terminal 2
1.2.4. Crearea și editarea în Visual Studio Code 3
1.3. Lucrul cu fișiere XML 4
1.3.1. Crearea fișierului XML 4
1.3.2. Adăugarea fișierului XML sub controlul versiunilor 4
1.4. Teme 5
1.4.1. Tema 1 5
1.4.2. Tema 2 5
1.4.3. Tema 3 5

1.1. Concepte și tehnologii

În acest laborator veți învăța să lucrați cu sistemul de versionare git și cu aplicația Visual Studio Code.

De asemenea, veți crea un document XML care reprezintă o grupare de persoane.

1.2. Crearea proiectului inițial

În cadrul laboratorului veți folosi sistemul de versionare git pentru a avea o evidență a modificărilor pe care le veți aduce asupra proiectelor. Mai multe informații cu privire la utilizarea unui sistem de versionare se găsesc în documentul **Cursul XX**.

Va trebui să vă creați cont pe site-ul https://github.com/ folosind adresa instituțională de forma @student.tuiasi.ro sau puteți folosi contul personal. În cele ce urmează sunt prezentați pașii care trebuie parcurși pentru a crearea contului și crearea proiectului inițial.

1.2.1. Crearea contului pe GitHub cu adresa instituțională

Accesați site-ul https://github.com și creați-vă cont cu adresa voastră de email TUIASI. După care logați-vă în cont.

1.2.2. Crearea proiectului pe site-ul GitHub

Din browser accesați site-ul `https://classroom.github.com/a/`*`COD`*, unde *`COD`* este un șir de caractere care va fi comunicat în cadrul laboratorului. Atenție! Acest *`COD`* este unic pentru fiecare grupă.

Accesând adresa de mai sus veți crea un proiect (repository) pe site-ul GitHub care va putea fi monitorizat de cadrele didactice de la laborator.

Pe pagina care se deschide trebuie să apăsați butonul verde `Accept this assignment`.

Se va crea un proiect care este accesibil la adresa `https://github.com/HDR-2023-2024/proiect-1-`*`numeUtilizatorGitHub`*.

La accesarea adresei de mai sus aveți pașii necesari creării unui repository. Modul de utilizare a instrucțiunilor respective este prezentat în cele ce urmează.

1.2.3. Utilizarea Git din terminal

Deschideți Visual Studio Code, apăsați combinația de taste CTRL + ` (backtick) pentru a accesa terminalul.

Executați pe rând următoarele comenzi și, dacă întâmpinați erori, consultați informațiile de după comenzi:

```
# intră în directorul ~ (/home/student/)
cd ~

# creează un director cu numele proiectului (proiect-1-numeUtilizatorGitHub)
mkdir proiect-1-numeUtilizatorGitHub

# intră în directorul respectiv
cd proiect-1-numeUtilizatorGitHub

# creează un fișier README.md cu textul: # proiect-1-numeUtilizatorGitHub
echo "# proiect-1-numeUtilizatorGitHub" >> README.md

# creează un repository git local în directorul curent
git init

# afișează lista de fișiere care au modificări ce vor fi procesate la următorul commit
# ar trebui să apară: Untracked files: README.md
git status

# adaugă starea CURENTĂ a fișierului specificat la index (aceste modificări vor fi
procesate la următorul commit); operația se cheamă "to stage"
# înaintea unui commit, se poate apela de mai multe ori git add pentru același fișier sau
pentru fișiere diferite
git add README.md

# afișează lista de fișiere care au modificări ce vor fi procesate la următorul commit
# ar trebui să apară: Changes to be committed: new file: README.md
git status

# setează numele și adresa de email a persoanei care va face commit-urile pe acest
repository (de la calculatorul curent)
```

```
git config user.name "Prenume Nume"
git config user.email "numeUtilizatorTUIASI@student.tuiasi.ro"

# creează un nou commit care conține toate modificările din index (tot ce a fost adăugat cu git add)
git commit -m "primul commit"

# creează un remote (repository aflat în altă parte, e.g., la GitHub) cu numele origin care referă repository-ul de la GitHub
# este un fel variabilă cu numele origin care referă url-ul repository-ului de la GitHub.
# această instrucțiune trebuie apelată o singură dată!
git remote add origin https://github.com/HDR-2023-2024/proiect-1-numeUtilizatorGitHub.git

# dacă ați setat greșit origin-ul se poate șterge cu ajutorul comenzii: git remote remove origin

# redenumește branch-ul principal în main
git branch -M main

# toate commit-urile din branch-ul main (branch-ul implicit) sunt trimise la repository-ul referit de origin
# toate modificările din repository-ul local sunt trimise la repository-ul de la GitHub
# va trebui să introduceți username-ul și parola contului de GitHub când dați push.
git push -u origin main
```

Dacă atunci când vă autentificați, vă apare o eroare de forma **Support for password authentication was removed. Please use a personal access token instead**, atunci trebuie să utilizați un access token în loc de parolă. Acesta se creează astfel: de pe siteul github.com se apasă în partea din dreapta-sus unde este profilul utilizatorului autentificat ➜ Settings ➜ Developer settings (din meniul din stânga, ultimul item) ➜ Personal access tokens ➜ Generate new token ➜ la Note puneți o denumire pentru token ➜ bifați repo ➜ apoi, apăsați Generate token. Tokenul va fi folosit în loc de parolă și trebuie salvat undeva pentru că nu va mai fi afișat pe github.com.

În Windows, dacă, atunci când dați `git push` apare cum că nu se găsește repository-ul, iar numele acestuia este scris corect, cel mai probabil aveți două conturi de GitHub. Ca să rezolvați problema trebuie să apăsați tasta Windows scrieți Credential și lansați în execuție aplicația Credential Manager, apoi apăsați pe Windows Credentials și ștergeți intrările care conțin github.com din lista care apare. Apoi, dați din nou `git push` și urmați instrucțiunile care apar pe ecran.

În acest moment ar trebui ca fișierul README.md să apară la adresa `https://github.com/HDR-2023-2024/proiect-1-`*`numeUtilizatorGitHub`*.

1.2.4. Crearea și editarea în Visual Studio Code

Din meniul de sus din Visual Studio Code: File ➜ Open folder ➜ selectați directorul `/home/student/proiect-1-`*`numeUtilizatorGitHub`*.

Pentru a vedea fișierele din directorul selectat se accesează fereastra Explorer prin apăsarea butonului de sub meniul File (sau, din meniul de sus din Visual Studio Code: View ➜ Explorer).

Pentru a crea fișiere fișiere/directoare noi se apasă click dreapta în fereastra Explorer și se alege opțiunea New File / New Folder.

Extensiile pentru Visual Studio Code se pot instala din meniul de sus: View ➜ Extensions.

De exemplu, se poate instala XML 0.24.0 (redhat.vscode-xml) pentru lucrul cu fișiere XML. După instalare trebuie repornit Visual Studio Code.

Formatarea codului se realizează cu ALT + SHIFT + F (Windows) și CTRL + SHIFT + I (Linux).

1.3. Lucrul cu fișiere XML

Fișierele XML sunt utilizate pentru a stoca informație într-un mod ușor de citit pentru utilizator, dar și de o aplicație software. Detalii referitoare la structura unui fișier XML se găsesc în Cursul YY de pe Moodle și la adresa XML Tutorial - w3schools.com.

1.3.1. Crearea fișierului XML

Pentru început veți avea de realizat un fișier cu numele `persoane.xml` în cadrul proiectului git creat anterior. Acest fișier va conține informații referitoare la mai multe persoane.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<persoane>
    <persoana id="1">
        <nume>Popescu</nume>
        <prenume>Ion</prenume>
        <varsta>23</varsta>
    </persoana>
</persoane>
```

Validați codul XML de mai sus pe site-ul https://www.freeformatter.com/xml-validator-xsd.html

1.3.2. Adăugarea fișierului XML sub controlul versiunilor

În terminalul (CTRL + `) Visual Studio Code, asigurați-vă că sunteți în directorul `/home/student/proiect-1-`*`numeUtilizatorGitHub`*.

```
# afișează lista de fișiere care au modificări ce vor fi procesate la următorul commit
# ar trebui să apară: Untracked files: persoane.xml
git status

# adaugă starea CURENTĂ a fișierului specificat la index (aceste modificări vor fi
procesate la următorul commit); operația se cheamă "to stage"
# înaintea unui commit, se poate apela de mai multe ori git add pentru același fișier sau
pentru fișiere diferite
git add persoane.xml

# afișează lista de fișiere care au modificări ce vor fi procesate la următorul commit
# ar trebui să apară: Changes to be committed: new file: persoane.xml
git status

# creează un nou commit care conține toate modificările din index (tot ce a fost adăugat
cu git add)
```

```
git commit -m "adaugare fișier persoane.xml"

# toate commit-urile din branch-ul main (branch-ul implicit) sunt trimise la repository-ul referit de origin
# toate modificările din repository-ul local sunt trimise la repository-ul de la GitHub
git push -u origin main
```

1.4. Teme

1.4.1. Tema 1

Editați fișierul `persoane.xml` și adăugați un element cu numele `adresa` care să conțină elementele `strada`, `numar`, `localitate`, `judet` și `tara`.

Nu uitați să adăugați modificările sub controlul versiunilor și să validați xml-ul creat!

1.4.2. Tema 2

Editați fișierul `persoane.xml` și adăugați mai multe persoane. De asemenea, adăugați și alte proprietăți ale persoanelor (de exemplu, informațiile care apar în CV-ul europass).

1.4.3. Tema 3

Creați un fișier DTD sau XSD (la alegere) care să valideze XML-ul creat. Realizați validarea online.

Pentru validarea cu DTD din Visual Studio Code nu trebuie să aveți în DTD la început `<!DOCTYPE`. Validarea se face automat dacă aveți instalată extensia XML 0.24.0 (redhat.vscode-xml).

Tutorial DTD - https://www.w3schools.com/xml/xml_dtd_intro.asp

Tutorial XSD - https://www.w3schools.com/xml/schema_intro.asp

Programare web

Laboratorul 02

2. Crearea paginilor web

Cuprins

2. Crearea paginilor web 1
2.1. Concepte și tehnologii 1
2.2. Resurse utile 1
2.3. Teme 2
2.3.1. Tema 1 2
2.3.2. Tema 2 2
2.3.3. Tema 3 3

2.1. Concepte și tehnologii

În acest laborator veți crea o pagină web folosind doar limbajul HTML.

De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

Pentru a vizualiza mai ușor pagina web din Visual Studio Code puteți instalați extensia `open in browser` pentru Visual Studio Code:

- Din meniul de sus a Visual Studio Code: View ➔ Extensions.
- Scrieți `open in browser` și instalați extensia **open in browser v2.0.0** (`techer.open-in-browser`).

O dată ce este deschisă în browser, puteți observa modificările aduse asupra paginii web apăsând tasta F5 (Refresh) în browser sau combinația CTRL+F5 (Refresh cu golirea cache-ului).

2.2. Resurse utile

Pentru detalii privind limbajul HTML puteți să citiți cursul 1 și/sau să consultați documentația următoare:

- Specificațiile HTML – Living Standard - https://html.spec.whatwg.org/multipage/
- Tutorial w3schools - http://www.w3schools.com/html/default.asp
- MDN - Mozilla Developer Network - https://developer.mozilla.org/en-US/docs/Web/HTML
- Dive into HTML 5 - http://diveintohtml5.info/

Cheat sheet:

- HTML CheatSheet - https://htmlcheatsheet.com/
- HTML cheat sheet extins pentru începători (organizat pe categorii) https://edu.tuiasi.ro/pluginfile.php/134242/mod_folder/content/0/html-cheat-sheet-extended.pdf
- HTML5 cheat sheet compact https://edu.tuiasi.ro/pluginfile.php/134242/mod_folder/content/0/htmlcheatsheet.pdf

2.3. Teme

2.3.1. Tema 1

Dacă nu folosiți calculatorul propriu, urmați pașii din Laboratorul 0 pentru clona proiectul din laboratorul precedent.

După clonarea proiectului, creați un fișier HTML cu numele `despre.html` care să arate ca în Fig. 2-1 (fără conturul negru).

Atenție! Pentru nota maximă 10 va trebui ca site-ul pe care îl veți realiza în cadrul primului proiect să aibă o anumită tematică, iar temele din cadrul fiecărui laborator trebuie adaptate temei de site alese. Astfel, pagina despre.html pe care o creați trebuie să conțină toate elementele care țin de html din imaginile de mai jos, dar acestea trebuie adaptate la tema aleasă.

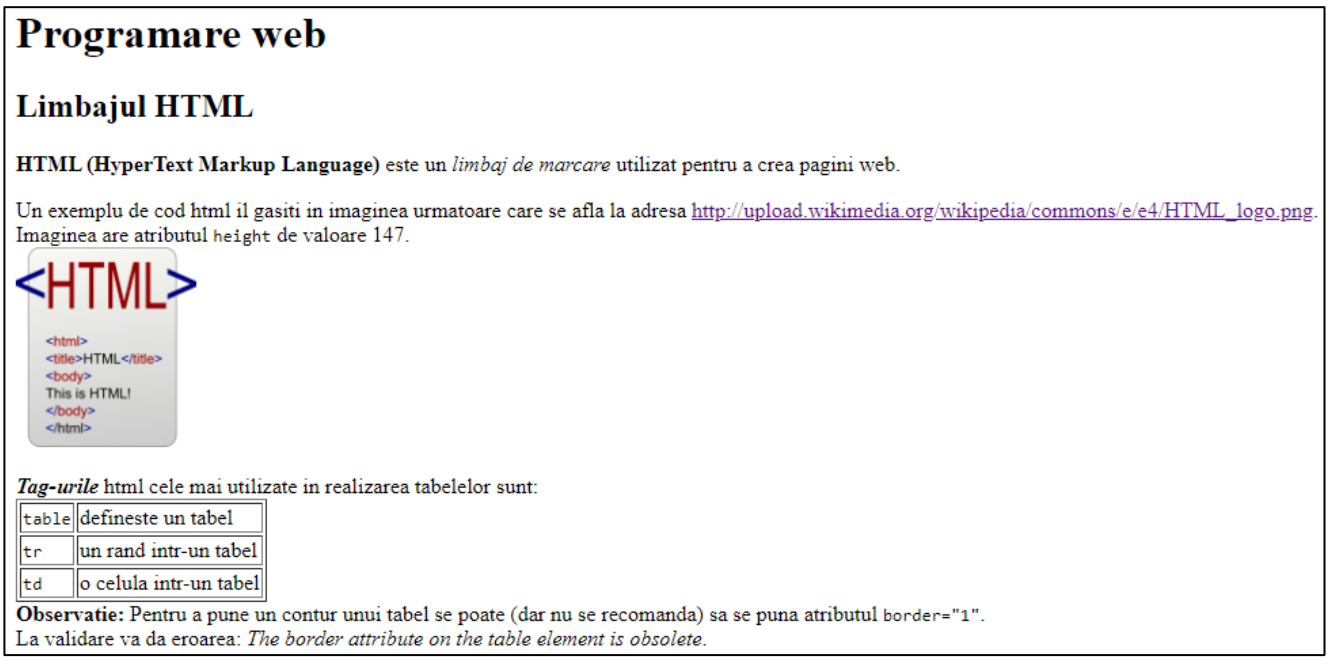


Fig. 2-1. Prima parte a fișierului despre.html

Testați fișierul în browser: din Visual Studio Code ➜ click dreapta pe fișierul `despre.html` ➜ selectați opțiunea Open in Default Browser.

2.3.2. Tema 2

Adăugați la documentul precedent conținut și tag-uri HTML astfel încât să arate ca în Fig. 2-2.

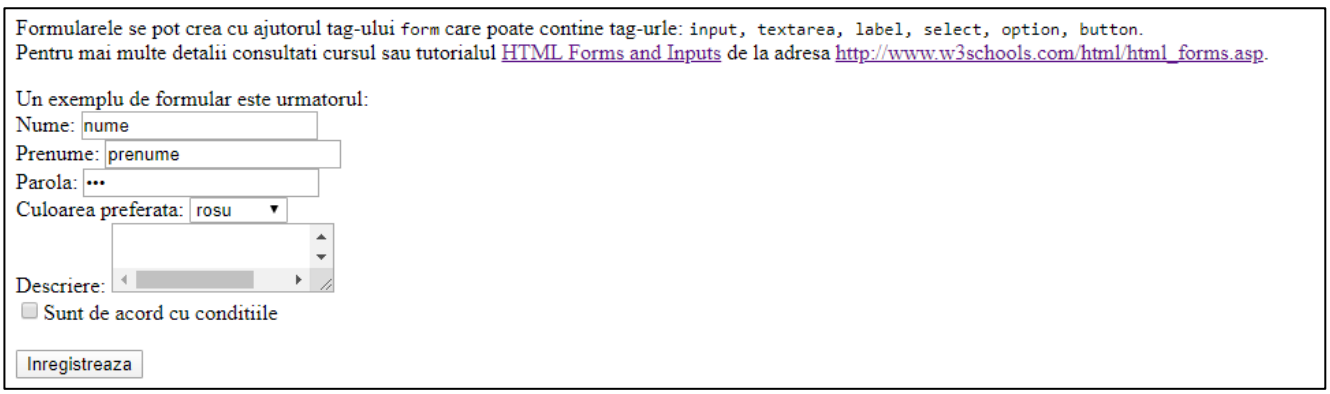


Fig. 2-2. A doua parte a fișierului despre.html

2.3.3. Tema 3

Adăugați la documentul precedent conținut și tag-uri HTML astfel încât să arate ca în Fig. 2-3.

Atenție la diacritice! Pentru a adăuga suport pentru limba română din Linux apăsați tasta windows ➜ scrieți region ➜ selectați Region & Language ➜ de la secțiunea Input sources apăsați pe + ➜ apăsați pe cele trei puncte verticale ➜ Romanian ➜ Romanian (standard cedilla) ➜ Add ➜ restart la calculator.

Validați documentul creat la adresa https://validator.w3.org/#validate_by_input.

Fig. 2-3. A treia parte a fișierului despre.html

Programare web

Laboratorul 03

3. Crearea paginilor web (2)

Cuprins

3. Crearea paginilor web (2) 1
3.1. Concepte și tehnologii 1
3.2. Resurse utile 1
3.3. Teme 1
3.3.1. Tema 1 1
3.3.2. Tema 2 2
3.3.3. Tema 3 2

3.1. Concepte și tehnologii

În acest laborator veți crea mai multe pagini web folosind doar limbajul HTML, cu accent pe elementele semantice.

De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

3.2. Resurse utile

Pentru detalii privind limbajul HTML puteți să citiți cursul 1 și/sau să consultați documentația următoare:

- Specificațiile HTML – Living Standard - https://html.spec.whatwg.org/multipage/
- Tutorial w3schools - http://www.w3schools.com/html/default.asp
- MDN - Mozilla Developer Network - https://developer.mozilla.org/en-US/docs/Web/HTML
- Dive into HTML 5 - http://diveintohtml5.info/
- HTML CheatSheet - https://htmlcheatsheet.com/
- W3C Markup Validation Service - https://validator.w3.org/#validate_by_input

3.3. Teme

3.3.1. Tema 1

Adăugați în proiectul de pe GitHub următoarele pagini web:

1. **Acasă** - fișierul `index.html` - va conține informații despre conținutul fiecărei secțiuni ale site-ului;
2. **Înregistrează** - fișierul `inregistreaza.html` - va permite crearea și completarea profilului personal al unui utilizator (vezi Tema 2);
3. **Video** - fișierul `video.html` - va conține mai multe fișiere video de pe YouTube (detalii referitoare la cum puteți să le adăugați în pagină găsiți pe pagina HTML YouTube Videos - https://www.w3schools.com/html/html_youtube.asp);
4. **Despre** - fișierul `despre.html` din laboratorul 2, adaptat ca să se integreze în site.

Observații:

- Pentru realizarea paginilor HTML folosiți tag-urile semantice de structurare a unei pagini web, și tag-urile și atributele specifice formularelor din HTML5.
- De asemenea, site-ul trebuie să conțină heading-uri, text formatat, imagini, link-uri, tabele, liste și formulare.
- Fiecare dintre cele patru pagini ale site-ului vor avea un meniu de navigare (același pentru toate paginile).
- Nu aveți voie să folosiți cod CSS.
- Documentul HTML va trebui scris folosind diacritice.
- Imaginile folosite trebuie stocate în directorul `imagini` care este localizat în același director cu fișierele html.
- Testați site-ul creat în Microsoft Edge, Firefox, Google Chrome și Opera pentru a observa diferențele dintre modul de afișare a elementelor HTML5.
- Validați paginile create la adresa http://validator.w3.org/.

3.3.2. Tema 2

Modificați site-ul descris mai sus astfel încât pagina Înregistrează să conțină un formular cu următoarele secțiuni/câmpuri (se va utiliza tag-ul fieldset):

- Informații generale:
  - Nume utilizator
  - Parola
  - Nume
  - Prenume
  - Email (atributul `email`)
  - Telefon (atributul `tel`)
- Informații personale:
  - Sexul (tag-urile `select/option`)
  - Mâncarea preferată (tag-urile `datalist/option`)
  - Culoarea preferată (atributul `color`)
  - Data nașterii (atributul `date`)
  - Ora nașterii (atributul `time`)
  - Vârsta (atributul `range`)
  - Adresa paginii personale (atributul `url`)
  - Descriere (tag-ul `textarea`)
- Un buton cu textul `Înregistrează` care va fi dezactivat.

3.3.3. Tema 3

Adăugați la site-ul de mai sus o pagină **Desen** care să conțină un SVG (Scalable Vector Graphics).

Desenul trebuie să conțină cel puțin cinci figuri geometrice diferite, text, gradienți și tranziții (vezi ultimul link de mai jos, la secțiunea SVG Misc).

Desenul trebuie să aibă o coerență și să nu fie doar o simplă copiere a exemplelor găsite la adresele de mai jos.

Informații cu privire la crearea și adăugarea unui desen SVG într-un fișier HTML găsiți la adresele:

- HTML SVG Graphics - https://www.w3schools.com/html/html5_svg.asp
- SVG Tutorial - https://www.w3schools.com/graphics/svg_intro.asp
- SVG Examples - https://www.w3schools.com/graphics/svg_examples.asp

Programare web

Laboratorul 04

4. Formatarea paginilor web

Cuprins

4. Formatarea paginilor web 1
4.1. Concepte și tehnologii 1
4.2. Structura proiectului 1
4.3. Resurse utile 1
4.4. Teme 2
4.4.1. Tema 1 2
4.4.2. Tema 2 2
4.4.3. Tema 3 2

4.1. Concepte și tehnologii

În acest laborator veți aplica stiluri folosind limbajul CSS.

De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

4.2. Structura proiectului

Structura proiectului `proiect1-numeUtilizatorGitHub` de până la acest laborator (inclusiv):

- [`css`]
  - `stil.css` - **lab 04**
- [`imagini`]
  - `logo.png` - lab 03
- `desen.html` - lab 03 tema 3, actualizat **lab 04**
- `despre.html` - lab 02, actualizat lab 03, **lab 04**
- `index.html` - lab 03, actualizat **lab 04**
- `inregistreaza.html` - lab 03, actualizat **lab 04**
- `persoane.dtd` și/sau `persoane.xsd` - lab 01
- `persoane.xml` - lab 01
- `video.html` - lab 03, actualizat **lab 04**

4.3. Resurse utile

Pentru detalii privind limbajul CSS puteți să citiți cursul 3 sau să consultați documentația următoare:

- Tutorial CSS - w3schools - https://www.w3schools.com/css/default.asp
- CSS: Cascading Style Sheets | MDN - https://developer.mozilla.org/en-US/docs/Web/CSS
- A Complete Guide to Flexbox - https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- A Complete Guide to CSS Grid - https://css-tricks.com/snippets/css/complete-guide-grid/

4.4. Teme

4.4.1. Tema 1

Adăugați stiluri CSS site-ului din proiectul de pe GitHub prin crearea unui fișier extern CSS cu numele `stil.css` în directorul `css` din rădăcina proiectului, atfel încât site-ul să arate bine.

Particularizați bara de meniu din elementul `nav` (fără a folosi elementele `ul` și `li`!) din toate paginile html (meniul trebuie să arate la fel pe toate paginile).

Observații:

- Trebuie pus accentul pe cum arată site-ul și pe accesul facil la conținutul acestuia.
- Pentru realizarea paginilor HTML folosiți tag-urile semantice și tag-urile și atributele specifice formularelor din HTML5.
- În realizarea layout-ului paginii folosiți stiluri CSS și nu tabele. Fiecare pagină trebuie să aibă același layout. Este de preferat să folosiți Flexbox în realizarea layout-lui (mai multe informații găsiți la HTML Layouts - http://www.w3schools.com/html/html_layout.asp).
- Fiecare dintre paginile site-ului vor avea un meniu de navigare (același pentru toate paginile). Eventual, anumite pagini pot avea un sub-meniu (tag-ul `aside`) pentru navigare în cadrul unei categorii.
- Documentul HTML va trebui scris folosind diacritice.
- Imaginile folosite trebuie stocate în directorul `imagini` care este localizat în același director cu fișierul `index.html`.
- Fișierele CSS externe folosite trebuie stocate în directorul `css` care este localizat în același director cu fișierul index.html.
- Testați site-ul creat în Edge/Safari, Firefox, Google Chrome și Opera pentru a observa diferențele dintre modul de afișare a elementelor HTML5.

4.4.2. Tema 2

Modificați site-ul și implementați următoarele caracteristici:

- Validați fișierele HTML și CSS și modificați-le astfel încât să nu apară warning-uri sau erori.
  - Validați fișierele HTML la adresa http://validator.w3.org/.
  - Validați fișierele CSS la adresa https://jigsaw.w3.org/css-validator/.
- Utilizați cel puțin o pseudo-clasă și un pseudo-element CSS.
- Formatați folosind CSS tabelul din pagina `despre.html`.
- Formatați folosind CSS formularul din pagina `inregistrare.html`.
- Adăugați în oricare pagină cel puțin 5 elemente specifice CSS3. (e.g., contururi, gradienți, efecte text, transformări 2D și 3D, animații, tranziții) - CSS Tutorial - https://www.w3schools.com/css/default.asp secțiunea CSS Advanced din meniul din dreapta)

4.4.3. Tema 3

Utilizați CSS Media Queries - https://www.w3schools.com/css/css3_mediaqueries.asp - pentru a face ca paginile să arate diferit în momentul în care este listată (printer friendly) și când este vizualizată pe dispozitive mobile - în special meniul din elementul `nav`.

Modificați pagina `video.html` cu ajutorul CSS Flexbox - https://www.w3schools.com/css/css3_flexbox.asp - astfel încât să aveți o galerie video în care video-urile au diferite dimensiuni.

Programare web

Laboratorul 05

5. Interacțiunea în paginile web

Cuprins

5. Interacțiunea în paginilor web 1
5.1. Concepte și tehnologii 1
5.2. Structura proiectului 1
5.3. Resurse utile 1
5.4. Teme 2
5.4.1. Tema 1 2
5.4.2. Tema 2 3
5.4.3. Tema 3 3

5.1. Concepte și tehnologii

În acest laborator veți utiliza limbajul JavaScript pentru a adăuga elemente de interacțiune a utilizatorului cu pagina web. De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

5.2. Structura proiectului

Structura proiectului `proiect1-numeUtilizatorGitHub` de până la acest laborator (inclusiv):

- [`css`]
  - `stil.css` - lab 04
- [`imagini`]
  - `logo.png` - lab 03
- [`js`]
  - `script.js` - **lab 05**
- `desen.html` - lab 03 tema 3, actualizat lab 04, **lab 05**
- `despre_html.html` - lab 02, actualizat lab 03, lab 04, **lab 05**
- `index.html` - lab 03, actualizat lab 04, **lab 05**
- `inregistreaza.html` - lab 03, actualizat lab 04, **lab 05**
- `invat.html` - **lab 05**
- `persoane.dtd` și/sau `persoane.xsd` - lab 01
- `persoane.xml` - lab 01
- `video.html` - lab 03, actualizat lab 04, **lab 05**

5.3. Resurse utile

Pentru detalii privind limbajul CSS puteți să citiți cursul 3 sau să consultați documentația următoare:

- Tutorial JavaScript - w3schools http://www.w3schools.com/js/default.asp
- JavaScript | MDN - https://developer.mozilla.org/en-US/docs/Web/JavaScript

5.4. Teme

5.4.1. Tema 1

Continuați proiectul de pe GitHub și:

a. creați un fișier `script.js` într-un director `js` din rădăcina proiectului,
b. creați un fișier invat.html în rădăcina proiectului,
c. modificați fișierele html create în laboratoarele precedente prin adăugarea în bara de navigare a unui link/buton textul "Învăț" și cu legătură spre fișierul `invat.html`,
d. adăugați secțiunile descrise mai jos în fișierul `invat.html` (partea de JavaScript trebuie să fie în `script.js`).

Observații:

- Documentul HTML va trebui scris folosind diacritice.
- Folosiți fișiere externe pentru CSS și JavaScript.
- Fișierele JavaScript externe folosite trebuiesc stocate în directorul `js` care este localizat în același director cu fișierul index.html.
- Documentul HTML nu trebuie doar să aibă funcționalitatea descrisă mai jos, ci trebuie să arate și bine.
- Pentru partea de JavaScript NU trebuie să se folosească biblioteci sau framework-uri.

Specificații generale:

- Pagina web (fișierul `invat.html`) trebuie să aibă aceleași elemente: header, nav și footer ca celelalte pagini din site.
- Toate secțiunile de mai jos trebuie adăugate în pagina `invat.html`.
- Nu este neapărat să faceți exact cum se cere în continuare. Adaptați cerințele de mai jos la tematica site-ului, dar și asigurați-vă că ați folosit toate elementele/conceptele referite în continuare.
- Elementul `aside` va conține un meniu cu legături către cele 3 secțiuni (tag-uri `section`) ale paginii.
- Pentru a realiza acest lucru accesați resursa HTML Links - The id Attribute[1].
- Fiecare secțiune va avea un heading (sugestiv) care să reprezinte titlul secțiunii respective.

Secțiunea 1:

- Prima secțiune este una informativă care va permite afișarea datei și timpului curent, a adresei URL, a locației curente, a numelui și a versiunii browser-ului, și a sistemului de operare folosit de utilizator.
- Pentru a obține informațiile enumerate anterior trebuie folosit evenimentul de `onload`[2], o funcție JavaScript care să fie apelată atunci când pagina web s-a încărcat, și o serie de proprietăți și metode ale obiectului `window`[3] din Browser Object Model (BOM).
- Ca să schimbați conținutul unui element HTML folosiți proprietatea `innerHTML` a metodei JavaScript `getElementById()`[4].
- Modalitatea de obținere a informațiilor referitoare la adresa URL o găsiți la documentația obiectului `window.location`[5], iar pentru locația curentă, browser și sistemul de operare găsiți la documentația obiectului `window.navigator`[6].
- În vederea afișării datei și a timpului curent se poate folosi obiectul Date[7].

[1] HTML Links – The id Attribute - http://www.w3schools.com/html/html_links.asp
[2] onload Event - https://www.w3schools.com/jsref/event_onload.asp
[3] The Window object - https://www.w3schools.com/jsref/obj_window.asp
[4] HTML DOM Document getElementById() - https://www.w3schools.com/jsref/met_document_getelementbyid.asp
[5] Window Location - https://www.w3schools.com/jsref/obj_location.asp
[6] Window Navigator - https://www.w3schools.com/jsref/obj_navigator.asp
[7] JavaScript Date Objects - https://www.w3schools.com/js/js_dates.asp

- Timpul trebuie actualizat în timp real, iar pentru a realiza acest lucru trebuie să folosiți evenimente de *timing*[8].

5.4.2. Tema 2

Modificați pagina web creată și adăugați secțiunea:

Secțiunea 2:

- A doua secțiune va permite desenarea unor figuri geometrice pe un `canvas`[9].
- Trebuie să existe posibilitatea de a selecta culoarea de desenare a conturului și a culorii de umplere (folosiți două color picker[10]-e).
- Desenarea pe canvas trebuie să se facă cu ajutorul mouse-ului (folosiți funcții JavaScript și JavaScript HTML DOM Events[11]).
- Este de ajuns să se deseneze doar dreptunghiuri; un dreptunghi este desenat prin apăsarea de două ori pe butonul mouse-ului în suprafața de desenare - coordonatele a două puncte diametral opuse ale dreptunghiului.
- Uitați-vă la exemplele de utilizare a funcțiilor `fill()` și `stroke()` pentru desenarea pe un canvas.
- Pentru a realiza integrarea în temă se poate folosi cod JavaScript care să se execute la încărcarea paginii pentru a exista deja imagini, text și/sau figuri geometrice peste care să deseneze utilizatorul.

5.4.3. Tema 3

Modificați pagina web creată și adăugați secțiunea:

Secțiunea 3:

- A treia secțiune va permite modificarea dinamică a unui tabel[12].
- Astfel, va exista posibilitatea de inserare a unei linii noi și a unei coloane noi la o poziție dată, precum și specificarea culorii de fundal a liniei, respectiv, coloanei inserate.
- În vederea implementării aveți nevoie de o căsuță text în care să specificați poziția (linia/coloana) din tabel la care se face inserarea, un color picker pentru specificarea culorii și două butoane pentru a insera o linie nouă, respectiv, o coloană nouă (folosiți tag-ul `input`).
- Pentru specificarea culorii de fundal folosiți CSS (CSS Background[13] și JavaScript HTML DOM - Changing CSS[14]).
- Funcționalitatea butoanelor trebuie dată prin JavaScript. Astfel, realizați câte o funcție pentru fiecare dintre cele două butoane, dați un `id` elementului `table` și folosiți metodele de navigare prin nodurile HTML prezentate în JavaScript HTML DOM Elements (Nodes)[15] și în JavaScript HTML DOM Node List[16].
- Atenție! Culoarea de fundal aplicați-o fiecărei celule a rândului/coloanei și nu întregului rând (întregii coloane nu aveți cum să specificați).
- Pentru a realiza integrarea în temă se poate porni de la un tabel cu informații referitoare la tema aleasă, iar utilizatorul să introducă linii și coloane noi de lățime/înălțime mai mică care doar să aibă rol de delimitarea a liniilor și a coloanelor existente.

[8] JavaScript Timing Events - https://www.w3schools.com/js/js_timing.asp
[9] HTML Canvas Reference - https://www.w3schools.com/tags/ref_canvas.asp
[10] HTML <input type="color"> - https://www.w3schools.com/tags/att_input_type_color.asp
[11] JavaScript HTML DOM Events - https://www.w3schools.com/js/js_htmldom_events.asp
[12] HTML <table> Tag - https://www.w3schools.com/tags/tag_table.asp
[13] CSS Backgrounds - https://www.w3schools.com/css/css_background.asp
[14] JavaScript HTML DOM - Changing CSS - https://www.w3schools.com/js/js_htmldom_css.asp
[15] JavaScript HTML DOM Elements (Nodes) - https://www.w3schools.com/js/js_htmldom_nodes.asp
[16] JavaScript HTML DOM Node Lists - https://www.w3schools.com/js/js_htmldom_nodelist.asp

Programare web

Laboratorul 06

6. Serverul web

Cuprins

6. Serverul web 1
6.1. Concepte și tehnologii 1
6.2. Structura proiectului 1
6.3. Modelul client-server 2
6.4. Protocolul HTTP 2
6.5. Teme 3
6.5.1. Tema 1 3
6.5.2. Tema 2 6
6.5.3. Tema 3 6

6.1. Concepte și tehnologii

În acest laborator veți utiliza limbajul Python sau Java pentru a crea un server web. De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

6.2. Structura proiectului

Structura proiectului `proiect1-numeUtilizatorGitHub` de până la acest laborator (inclusiv):

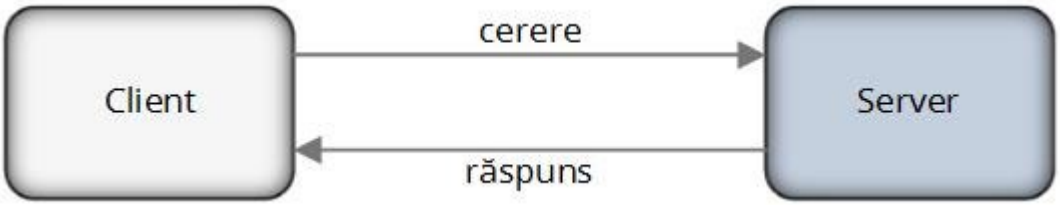
- [`server_web`]
  - `lanseaza_server.bat` (Windows) / `lanseaza_server.sh` (Linux) - **lab 06**
  - `server_web.py` (Python) / `ServerWeb.java` (Java) - **lab 06**
- [`continut`]
  - [`css`]
    - `stil.css` - lab 04
  - [`imagini`]
    - `logo.png` - lab 03
  - [`js`]
    - `script.js` - lab 05
  - `desen.html` - lab 03 tema 3, actualizat lab 04, lab 05
  - `despre.html` - lab 02, actualizat lab 03, lab 04, lab 05
  - `favicon.ico` - **lab 06**
  - `index.html` - lab 03, actualizat lab 04, lab 05
  - `inregistreaza.html` - lab 03, actualizat lab 04, lab 05
  - `invat.html` - lab 05
  - `persoane.dtd` și/sau `persoane.xsd` - lab 01
  - `persoane.xml` - lab 01
  - `video.html` - lab 03, actualizat lab 04, lab 05

6.3. Modelul client-server

Server = instanță a unei aplicații care primește cereri și oferă răspunsuri.

Client = instanță care accesează serviciile puse la dispoziție de un server.

Serverul așteaptă ("ascultă") conexiuni de la potențiali clienți. Clientul se conectează la server și trimite mesaje (cereri), iar serverul răspunde.



Comunicarea dintre server și client se realizează prin intermediul socket-urilor.

Un **socket** reprezintă o instanță la unul din cele două capete ale unei căi de comunicații și este format din IP și port.

IP-ul este o adresă de Internet a calculatorului și reprezintă o etichetă numerică asignată unui dispozitiv conectat la Internet (e.g., 192.168.1.101).

Port-ul este identificatorul unui capăt al unei căi de comunicații. Este folosit pentru a identifica în mod unic o anumită aplicație sau un serviciu care se execută pe un calculator în vederea comunicării printr-o rețea de calculatoare. (e.g., 8080).

Mai multe informații privind modelul client-server se găsesc în cursul 5.

6.4. Protocolul HTTP

Protocolul HTTP are la bază modelul client-server, iar în varianta cea mai simplistă comunicarea, din perspectiva serverului, este în felul următor:

1. Serverul este pornit pe un anumit port și așteaptă conexiuni din partea clienților (e.g., browser web),
2. La server se conectează un client,
3. Serverul primește cererea HTTP de la un client (i.e., mai multe linii de text -> vezi în continuare cererea HTTP),
4. Serverul interpretează cererea (i.e., determină ce resursă este cerută),
5. Serverul trimite răspunsul (i.e., mai multe linii de text + conținutul resursei cerute -> vezi în continuare răspunsul HTTP),
6. Serverul închide conexiunea cu clientul.

Exemplu practic: Dacă în browser se introduce adresa (URI-ul) http://localhost:5678/index.html, atunci cererea și răspunsul HTTP arată astfel:

Cererea HTTP este de forma:

```
GET /index.html HTTP/1.1
Host: localhost:5678
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:32.0) Gecko/20100101 Firefox/32.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Răspunsul HTTP trebuie să conțină următoarele secțiuni separate de CRLF (Carriage Return - Line Feed) (`\r\n`):

1. linia de start = `HTTP/1.1 200 OK`, dacă resursa cerută reprezintă un fișier valid, sau `HTTP/1.1 404 Not Found`, dacă nu este un fișier valid.
2. header-e (perechi cheie-valoare) - liniile header din răspuns ar trebui să conțină cel puțin:
   - `Content-Length`: lungimea_corpului_mesajului_de_răspuns
   - `Content-Type`: tipul_răspunsului (e.g., text/plain)
   - `Server`: numele_serverului_vostru
3. CRLF (`\r\n`)
4. Corpul mesajului de răspuns va fi conținutul fișierului reprezentat de resursa cerută.

Mai multe informații privind protocolul HTTP se găsesc în cursul 5.

6.5. Teme

6.5.1. Tema 1

Creați un server web care răspunde cu textul **Hello World! - *numele resursei cerute***. Clientul va fi browser-ul: http://localhost:5678/index.html. La accesarea acestei pagini web, browser-ul va face o cerere la server pentru fișierul `index.html`, apoi încă o cerere pentru fișierul `favicon.ico`.

Aveți de ales între o implementare în Python, una în Java sau una în orice alt limbaj.

Pentru a realiza acest lucru, continuați proiectul de pe GitHub și efectuați următorii pași (după fiecare pas, dați `git add` și `commit`):

a. creați un director cu numele `continut`, și mutați toate fișierele și celelalte directoare din rădăcina proiectului în directorul nou creat,
b. creați în directorul `continut` un fișier cu numele `favicon.ico` care reprezintă iconița site-ului vostru (puteți folosi site-urile https://www.favicon.cc, https://favicon.io),
c. creați un director cu numele `server_web` în rădăcina proiectului,
d. în directorul `server_web` creați fișierul `server_web.py` (Python) / `ServerWeb.java` (Java) - în continuare aveți o bază de pornire,
e. în directorul `server_web` creați fișierul `lanseaza_server.bat` (Windows) / `lanseaza_server.sh` (Linux) - care reprezintă scriptul de execuție a serverului web:
   - Python: `c:\Python27\python.exe server_web.py`
   - Java: `javac ServerWeb.java` și `java ServerWeb`

Pe browser-ele mai noi trebuie să fie trimis răspunsul în format HTTP (inclusiv linia de start și headere-e) după cum este explicat mai sus.

Serverul web - Varianta Python

```
import socket

# creeaza un server socket
serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# specifica ca serverul va rula pe portul 5678, accesibil de pe orice ip al serverului
serversocket.bind(('', 5678))
# serverul poate accepta conexiuni; specifica cati clienti pot astepta la coada
serversocket.listen(5)
```

```
while True:
    print
'################################################################################'
    print 'Serverul asculta potentiali clienti.'
    # asteapta conectarea unui client la server
    # metoda `accept` este blocanta => clientsocket, care reprezinta socket-ul
corespunzator clientului conectat
    (clientsocket, address) = serversocket.accept()
    print 'S-a conectat un client.'
    # se proceseaza cererea si se citeste prima linie de text
    cerere = ''
    linieDeStart = ''
    while True:
        data = clientsocket.recv(1024)
        cerere = cerere + data.decode()
        print 'S-a citit mesajul: \n---------------------------\n' + cerere + '\n-
---------------------------'
        pozitie = cerere.find('\r\n')
        if (pozitie > -1):
            linieDeStart = cerere[0:pozitie]
            print 'S-a citit linia de start din cerere: ##### ' + linieDeStart + '
#####'
            break
    print 'S-a terminat cititrea.'
    # TODO interpretarea sirului de caractere `linieDeStart` pentru a extrage
numele resursei cerute
    # TODO trimiterea răspunsului HTTP
    clientsocket.close()
    print 'S-a terminat comunicarea cu clientul.'
```

Alte precizări:

- Pentru interpretarea liniei de start se poate folosi metoda `split`.
- Pentru trimiterea unui șir de caractere se poate folosi: `clientsocket.sendall(mesaj).`
- Pentru a determina lungimea unui șir de caractere și transformarea lungimii în șir se poate folosi: `str(len(mesaj.encode('utf-8'))).`

Serverul web - Varianta Java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class ServerWeb {
    public static void main(String[] args) throws IOException {
```

```
System.out.println("##################################################################
############");
        System.out.println("Serverul asculta potentiali clienti.");
        // pornește un server pe portul 5678
        ServerSocket serverSocket = new ServerSocket(5678);
        while(true) {
            // așteaptă conectarea unui client la server
            // metoda accept este blocantă
            // clientSocket - socket-ul clientului conectat
            Socket clientSocket = serverSocket.accept();
            System.out.println("S-a conectat un client.");
            // socketWriter - wrapper peste fluxul de ieșire folosit pentru a
transmite date clientului
            PrintWriter socketWriter = new
PrintWriter(clientSocket.getOutputStream(), true);
            // socketReader - wrapper peste fluxul de intrare folosit pentru a
primi date de la client
            BufferedReader socketReader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

            // este citită prima linie de text din cerere
            String linieDeStart = socketReader.readLine();
            System.out.println("S-a citit linia de start din cerere: ##### " +
linieDeStart + " #####");
            // mesajul citit este transmis la client
            # TODO interpretarea sirului de caractere `linieDeStart` pentru a
extrage numele resursei cerute
            # TODO trimiterea răspunsului HTTP
            // închide conexiunea cu clientul
            // la apelul metodei close() se închid automat fluxurile de intrare și
ieșire (socketReader și socketWriter)
            clientSocket.close();
            System.out.println("S-a terminat comunicarea cu clientul.");
        }
        // închide serverul
        //serverSocket.close();
    }
}
```

Alte precizări:

- Pentru interpretarea liniei de start se poate folosi clasa `java.lang.String` [1]cu metoda `split()` ca în exemplu[2].
- Pentru trimiterea unui șir de caractere se poate folosi: `socketWriter.print(`*`mesaj`*`)` pentru șiruri de caractere sau `clientSocket.getOutputStream().write(`*`octeți`*`)` pentru octeți.
- Pentru a determina lungimea unui șir de caractere se poate folosi: *`mesaj`*`.length()`.

[1] String - http://docs.oracle.com/javase/7/docs/api/java/lang/String.html
[2] Java - String split() Method - https://www.tutorialspoint.com/java/java_string_split.htm

6.5.2. Tema 2

Extindeți serverul web astfel încât serverul să trimită ca răspuns conținutul fișierului din directorul `continut` corespunzător resursei cerute de către browser.

- Pentru aceasta trebuie să citiți octeții din fișierul corespunzător resursei cerute, și să îi trimiteți în corpul mesajului de răspuns.
- Tratați și situația în care resursa cerută nu există.
- Valoarea pentru header-ul `Content-Length` reprezintă dimensiunea fișierului corespunzător resursei cerute.
- Valoarea pentru header-ul `Content-Type` depinde de tipul resursei cerute, astfel:
    - html ➜ text/html
    - css ➜ text/css
    - js ➜ application/js
    - png ➜ text/png
    - jpg sau jpeg ➜ text/jpeg
    - gif ➜ text/gif
    - ico ➜ image/x-icon

6.5.3. Tema 3

Extindeți serverul web și:

- Adăugați suport în răspunsul formulat la server pentru header-ul `Content-Encoding: gzip`. Acest lucru presupune compresarea corpului mesajului de răspuns.
- Faceți ca serverul să poată să proceseze cereri în paralel pentru mai mulți clienți.

Programare web

Laboratorul 07

7. Comunicarea asincronă

Cuprins

7. Comunicarea asincronă 1
7.1. Concepte și tehnologii 1
7.2. Structura proiectului 1
7.3. Resurse utile 2
7.4. Teme 2
7.4.1. Tema 1 2
7.4.2. Tema 2 3
7.4.3. Tema 3 4

7.1. Concepte și tehnologii

În acest laborator veți utiliza și extinde serverul web din laboratorul precedent și veți comunica asincron cu serverul prin intermediul AJAX.. De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

7.2. Structura proiectului

Structura proiectului `proiect1-numeUtilizatorGitHub` de până la acest laborator (inclusiv):

- [`server_web`]
  - `lanseaza_server.bat` (Windows) / `lanseaza_server.sh` (Linux) - lab 06
  - `server_web.py` (Python) / `ServerWeb.java` (Java) - lab 06, actualizat **lab 07**
- [`continut`]
  - [`css`]
    - `stil.css` - lab 04
  - [`imagini`]
    - `logo.png` - lab 03
  - [`js`]
    - `script.js` - lab 05
    - `persoane.js`- **lab 07**
  - [`resurse`]
    - `persoane.dtd` și/sau `persoane.xsd` - lab 01
    - `persoane.xml` - lab 01
    - `utilizatori.json` - **lab 07**
  - `acasa.html` - **lab 07**
  - `desen.html` - lab 03 tema 3, actualizat lab 04, lab 05**, lab07**
  - `despre.html` - lab 02, actualizat lab 03, lab 04, lab 05**, lab07**
  - `favicon.ico` - lab 06

- `index.html` - lab 03, actualizat lab 04, lab 05**, lab07**
- `inregistreaza.html` - lab 03, actualizat lab 04, lab 05**, lab07**
- `invat.html` - lab 05**, lab07**
- `persoane.html` - **lab07**
- `verifică.html` - **lab07**
- `video.html` - lab 03, actualizat lab 04, lab 05**, lab07**

7.3. Resurse utile

Pentru detalii privind limbajul CSS puteți să citiți cursurile 4, 5 și 6, sau să consultați documentația următoare:

a. Tutorial JavaScript - http://www.w3schools.com/js/default.asp
b. JavaScript | MDN - https://developer.mozilla.org/en-US/docs/Web/JavaScript
c. Tutorial AJAX - https://www.w3schools.com/xml/ajax_intro.asp

Serverul web din laboratorul precedent:

a. Server web Python 2 - https://edu.tuiasi.ro/mod/resource/view.php?id=32608
b. Server web Python 3 - https://edu.tuiasi.ro/mod/resource/view.php?id=32614
c. Server web Java - https://edu.tuiasi.ro/mod/resource/view.php?id=32611

7.4. Teme

7.4.1. Tema 1

Modificați proiectul de pe GitHub astfel încât site-ul să devină un Single Page Application[1].

Accesul la paginile web ale site-ului se va realiza prin intermediul paginii `index.html`, care va conține scheletul site-ului (tot mai puțin conținutul elementului `<main>`).

Prin intermediul AJAX se va încărca în mod dinamic conținutul paginilor *Acasă, Desen, Despre HTML, Înregistrează, Învăț JS, Persoane, Verifică* și *Video*.

Pentru a realiza acest lucru, continuați proiectul de pe GitHub și efectuați următorii pași (după fiecare pas, dați git add și commit):

a. creați în directorul `continut` un fișier cu numele `acasa.html` în care copiați tot conținutul fișierului `index.html`,
b. în fișierul `index.html`, ștergeți tot conținutul elementului `<main>` și adăugați-i atributul `id="continut"`,
c. în fișierul `index.html`, adăugați o referință la fișierul `script.js`,
d. modificați conținutul celorlalte fișiere html astfel încât să rămână în fișierele respective doar conținutul elementului `<main>` (ștergeți tot codul de deasupra tag-ului `<main>` și de după tag-ul `</main>`, inclusiv cele două tag-uri),
e. în fișierul `script.js` adăugați o funcție cu numele `schimbaContinut` care primește ca argument un șir de caractere cu numele resursa (în funcția respectivă se va folosi AJAX pentru a se face cerere pentru resursa `resursa + '.html'`, iar răspunsul va fi setat ca și conținut al elementului cu id-ul `continut`),
f. în fișierul `index.html`, modificați link-urile interne prin înlăturarea atributului `href` și adăugarea atributului `onclick` (valoarea acestui atribut va fi apelul funcției `schimbaContinut`, iar ca argument va fi numele resursei cerute; e.g., `<a onclick="schimbaContinut('acasa')">Acasă</a>`).

[1] Single Page Application - https://developer.mozilla.org/en-US/docs/Glossary/SPA

g. de acum, toate fișierele html exceptând `index.html` vor fi încărcate folosind AJAX și nu accesate direct din browser; utlizatorul nu părăsește pagina `index.html`, ci doar conținutul elementului `<main>` se schimbă,
h. de asemenea, adăugați la server suport pentru fișiere xml și json prin adăugarea logicii corespunzătoare la header-ul `Content-Type`: json ➜ application/json, xml ➜ application/xml.

7.4.2. Tema 2

Extindeți proiectul prin adăugarea fișierului `persoane.html` în rădăcina proiectului. Pe această pagină se va permite vizualizarea sub forma unui tabel a persoanelor din fișierul `persoane.xml` creat în primul laborator.

a. Pentru început, mutați fișierul `persoane.xml` și fișierele `persoane.dtd` / `persoane.xsd` (dacă există), într-un director cu numele resurse din directorul `continut`.
b. Prin AJAX, la apăsarea butonului `Persoane`, din index.html se cere resursa `persoane.html`. După ce s-a primit răspunsul pentru persoane.html ar trebui, să se execute un script care să ceară fișierul `persoane.xml`. Problema este că nu se execută niciun cod JavaScript din interiorul fișierului `persoane.html`.
c. Va trebui să modificați funcția `schimbaContinut` ca să primească încă două argumente (`jsFisier` - reprezintă numele fișierului JavaScript care se va încărca după încărcarea resursei cerute, `jsFunctie` - reprezintă numele funcției care se va apela după ce se încarcă scriptul din `jsFisier`) și să adăugați când se primește răspunsul, după linia

```
document.getElementById("continut").innerHTML = this.responseText;
```

, următorul cod:

```
if (jsFisier) {
    var elementScript = document.createElement('script');
    elementScript.onload = function () {
        console.log("hello");
        if (jsFunctie) {
            window[jsFunctie]();
        }
    };
    elementScript.src = jsFisier;
    document.head.appendChild(elementScript);
} else {
    if (jsFunctie) {
        window[jsFunctie]();
    }
}
```

În bara de navigare codul pentru butonul `Persoane` este de forma:

```
<a onclick="schimbaContinut('persoane', 'js/persoane.js', 'incarcaPersoane')">Persoane</a>
```

d. Conținutul fișierului `persoane.html` va fi doar textul *Se încarcă. Vă rugăm așteptați....*
e. În fișierul persoane.js va exista funcția `incarcaPersoane` care conține logica prin care se face o cerere la server folosind AJAX pentru fișierul `persoane.xml`.

f. În momentul primirii fișierului XML se vor folosi funcții specifice DOM pentru a parcurge XML-ul (XML DOM[2]) și a genera un tabel HTML.
g. Tabelul va fi afișat în secțiune în locul mesajului de încărcare și va avea stiluri CSS.

7.4.3. Tema 3

Extindeți proiectul prin adăugarea fișierului verifica.html în rădăcina proiectului. Pe această pagină se va permite verificarea dacă un nume de utilizator și parola acestuia sunt introduse corect.

a. Pentru început, creați un fișier `utilizatori.json` în directorul resurse care va conține codul:
```
[{ "utilizator": "test", "parola": "test"}]
```
b. În fișierul `verifica.html` trebuie să creați de două căsuțe text, un buton și un element în care să se afișeze dacă utilizatorul și parola sunt corecte, și o funcție JavaScript în care să se facă validarea.
c. La apăsarea butonului, prin intermediul AJAX, se va lua de la server un fișier JSON cu numele `utilizatori.json`, iar prin intermediul JavaScript se va verifica dacă numele utilizatorului se află în lista din JSON și dacă corespunde parola (JavaScript JSON[3]). Folosiți metoda `JSON.parse(`*`text`*`)` pentru a converti un text care reprezintă un JSON la obiect.
d. Extindeți funcționalitatea formularului din fișierul `inregistreaza.html` creat în laboratorul 3, astfel încât, la apăsarea butonului Înregistrează, să se facă o cerere prin metoda HTTP POST pentru resursa `/api/utilizatori`. La server, când se primește o astfel de cerere se interpretează corpul mesajului, informațiile de acolo sunt transformate în obiect JSON, și apoi este actualizat corespunzător fișierul `resurse/utilizatori.json` de la server.
   Atenție! Nu trebuie să existe vreun fișier `/api/utilizatori` la server, ci doar se verifică dacă resursa cerută de client este `/api/utilizatori` și apoi se face procesarea respectivă.

[2] XML DOM - https://www.w3schools.com/xml/xml_dom.asp
[3] JavaScript DOM - https://www.w3schools.com/js/js_json.asp

Programare web

Laboratorul 08

8. Web API

Cuprins

8. Web API 1
8.1. Concepte și tehnologii 1
8.2. Structura proiectului 1
8.3. Resurse utile 2
8.4. Teme 2
8.4.1. Tema 1 2
8.4.2. Tema 2 3
8.4.3. Tema 3 3

8.1. Concepte și tehnologii

În acest laborator veți lucra cu concepte precum prototipuri/clase, Promises, Web Storage, Web Worker, IndexedDB. De asemenea, veți lucra în continuare cu sistemul de versionare git și cu aplicația Visual Studio Code.

8.2. Structura proiectului

Structura proiectului `proiect1-`*`numeUtilizatorGitHub`* de până la acest laborator (inclusiv):

- [`server_web`]
  - `lanseaza_server.bat` (Windows) / `lanseaza_server.sh` (Linux) - lab 06
  - `server_web.py` (Python) / `ServerWeb.java` (Java) - lab 06, actualizat lab 07
- [`continut`]
  - [`css`]
    - `stil.css` - lab 04
  - [`imagini`]
    - `logo.png` - lab 03
  - [`js`]
    - `cumpărături.js` - **lab 08**
    - `script.js` - lab 05
    - `persoane.js`- lab 07
    - `worker.js` - **lab 08**
  - [`resurse`]
    - `persoane.dtd` și/sau `persoane.xsd` - lab 01
    - `persoane.xml` - lab 01
    - `utilizatori.json` - lab 07
  - `acasa.html` - lab 07
  - `cumparaturi.html` - **lab 08**
  - `desen.html` - lab 03 tema 3, actualizat lab 04, lab 05, lab07

- `despre.html` - lab 02, actualizat lab 03, lab 04, lab 05, lab07
- `favicon.ico` - lab 06
- `index.html` - lab 03, actualizat lab 04, lab 05, lab 07, **lab08**
- `inregistreaza.html` - lab 03, actualizat lab 04, lab 05, lab07
- `invat.html` - lab 05, lab07
- `persoane.html` - lab07
- `verifică.html` - lab07
- `video.html` - lab 03, actualizat lab 04, lab 05, lab07

8.3. Resurse utile

Pentru detalii privind limbajul CSS puteți să citiți cursurile 4, 5 și 6, sau să consultați documentația următoare:

a. Tutorial JavaScript - http://www.w3schools.com/js/default.asp
b. JavaScript | MDN - https://developer.mozilla.org/en-US/docs/Web/JavaScript
c. Tutorial AJAX - https://www.w3schools.com/xml/ajax_intro.asp
d. Using the Web Storage API - https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API
e. Using Web Workers - https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers
f. Using IndexedDB - https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Using_IndexedDB

8.4. Teme

8.4.1. Tema 1

Modificați proiectul de pe GitHub astfel încât să existe posibilitatea creării unei liste de cumpărături.

Adăugarea produselor se va face din pagina `cumparaturi.html`, care va fi integrată în site-ul existent.

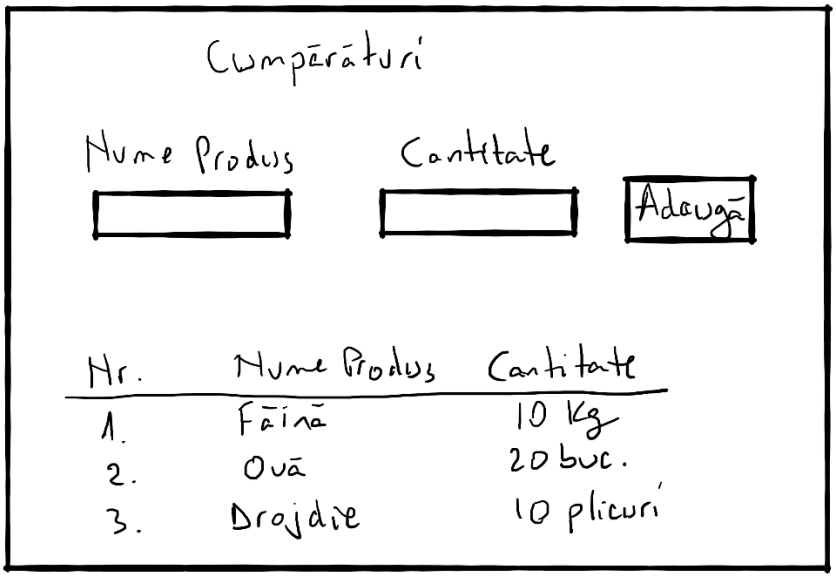
Pentru a realiza acest lucru, continuați proiectul de pe GitHub și efectuați următorii pași (după fiecare pas, dați git add și commit):

- Creați în directorul `continut` un fișier cu numele `cumparaturi.html` în care trebuie să aveți un formular cu minim două căsuțe text `nume` și `cantitate`, și un buton `Adaugă`,
- Creați în directorul `continut/js` un fișier cu numele `cumparaturi.js` care să conțină logica de adăugare a unui item la lista de cumpărături,
- Pentru fiecare item, creați un obiect de tipul `Produs` (utilizați funcții constructor[1] sau clase[2]),
- Unde se justifică folosiți promises[3] și arrow functions[4],
- De fiecare dată când se apasă butonul Adaugă, utilizați Web Storage API - localStorage[5] pentru a salva în browser produsul respectiv din lista de cumpărături (fiecare produs va conține și o propietate `id` care reprezintă numărul de ordine al produsului).

[1] JavaScript Object Constructors - https://www.w3schools.com/js/js_object_constructors.asp
[2] JavaScript Classes - https://www.w3schools.com/js/js_classes.asp
[3] Promises | MDN - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
[4] Arrow function expressions | MDN
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions
[5] Using the Web Storage API | MDN-
https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API

8.4.2. Tema 2

Extindeți proiectul prin adăugarea următorului comportament paginii `Cumpărături`, astfel încât să arate similar cu figura:



a. Creați un Web Worker[6] (fișierul `continut/js/worker.js`) care este notificat de script-ul principal de fiecare dată când s-a apăsat butonul `Adaugă`.
b. De fiecare dată când worker-ul este notificat, acesta va afișa un mesaj la consola browserului și va notifica scriptul care l-a creat (`cumparaturi.js`), care la rândul său va adăuga o linie în tabelul cu lista de cumpărături.

8.4.3. Tema 3

Extindeți proiectul și permiteți utilizatorului să aleagă modalitatea de salvare a listei de cumpărături și, anume, dacă să fie utilizat `localStorage` din Web Storage API sau să fie utilizat IndexedDB[7].

Creați câte o clasă pentru fiecare dintre cele două modalități de stocare, și ambele clase trebuie să moștenească aceeași clasă de bază prin intermediul `class extends` sau `prototype`.

[6] Using Web Workers - https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers
[7] Using IndexedDB - https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Using_IndexedDB

Programare web

Laboratorul 09

9. Prezentarea proiectului 1

În acest laborator veți avea de prezentat ce ați lucrat în primele opt săptămâni (i.e., proiectul 1).

Veți prezenta părți din cod, site-ul și repository-ul de pe git.

Sunt 15 criterii care sunt evaluate în ordinea specificată de cadrul didactic de la laborator.

Criteriile sunt în funcție de temele avute în fiecare laborator, aspectul site-ului și integrarea site-ului în tema aleasă.

Notele obținute la proiectul 1 vor fi comunicate, probabil, în săptămâna a 9-a. Notele nu se rotunjesc.