Proiect

DataSets Join

Pasi generali pentru realizarea proiectului:

1. Intelegerea datelor: examinarea fiecarui fisier CSV in parte pentru a intelege structura acestora, coloanele pe care le contin si datele aferente fiecarei coloane (daca acestea se afla pe coloana potrivita sau sunt decalate/amestecate).
2. Citirea fisierelor, convertirea fisierelor selectate si incarcarea acestora pentru realizarea fisierului final;
3. Identificarea coloanelor comune din cele trei fisiere, in cazul acesta, coloanele comune si de interes sunt: Domeniul, Adresa, Tara, Categoria, Telefonul si Numele Companiei.
4. Stabilirea coloanei pentru imbinarea celor trei fisiere. Pentru a unii fisierele am stabilit ca si cheie comuna, coloana “domain” (aferenta fisierelor Facebook_datasets.CSV si Google_datasets.CSV), respectiv coloana “root_domain” (aferenta fisierului Website.datasets.CSV).
5. Gestionarea conflictelor care pot aparea intre datele dintre cele 3 fisiere. Pot aparea conflicte reprezentate de discrepante intre valori pentru aceeasi entitate (de exemplu adrese sau numere de telefon diferite). Pentru a decide cum sa rezolvam aceste conflicte, trebuie sa analizam diversi factori, cum ar fi: calitatea datelor, fiabilitatea surselor sau frecventa de aparitie.
6. Generearea fisierului final pe baza conditiilor mentionate mai sus.
7. Verificarea si validarea datelor din fisierul final.

Explicatii proiect pe baza codului:

1. Codul furnizat este o pagina HTML cu denumirea “Join Datasets” care permite utilizatorului sa incarce mai multe fisiere de tip CSV, folosind un element de intrare (input) de tip fisier.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="./datasets_join.css">
<title>Join Datasets</title>
</head>
<body>
  <div class = "containerClass">
    <input type = "file" id = "fileInput" multiple accept = ".csv">
```

2. Script-urile incarcate sunt utilizate pentru a avea acces la librariile Node.js. Bibliotecile JavaScript “lodash” si “Paparse” ne permit manipularea si analiza datelor de tip CSV.

```
<script src = "https://cdn.jsdelivr.net/npm/lodash@4.17.21/lodash.min.js"></script>
<script src = "https://cdnjs.cloudflare.com/ajax/libs/PapaParse/5.3.0/papaparse.min.js"></script>
```

3. Dupa ce fisierele au fost incarcate cu success, utilizatorul va apasa apasa butonul “Merge CSV”, care va unii cele trei fisiere intr-un singur fisier si va permite descarcarea setului de date final in format CSV. Ori de cate ori utilizatorul va apasa pe buton, se va apela functia convertAndMergeCSV().

```
<button onclick = "convertAndMergeCSV()" > Merge CSV </button>
```

4. Functia convertAndMergeCSV() ne permite sa:
    - Preluam fisierele selectate de catre utilizator si sa accesam proprietatile acestora,
    - Pentru fiecare fisier CSV in parte, apeleaza functia readAndParseFile(), care are ca scop citirea si convertirea fiecarui fisier in parte intr-un array de obiecte.
        - Functia readAndParseFile() primeste ca si parametru un obiect de tip file, care reprezinta fisierul selectat.
        - Functia readAndParseFile() utilizeaza obiectul JS denumit FileReader() care permite citirea asincrona a datelor de catre o pagina web de pe computerul utilizatorului.
        - reader.onload este un event handler care este apelat ori de cate ori citirea fisierului s-a realizat cu success.
        - Cand fisierul este incarcat cu succes, aceasta functie va fie apelata cu un obiect eveniment care conţine informatii despre fisierul incarcat.
        - Functia readAndParseFile() utilizeaza “PapaParse” pentru a converti fisierul intr-un array de obiecte.
        - Dupa ce datele au fost citite si covertire, acestea sunt adaugate intr-un array, si anume: csvDataSets.
        - Daca toate fisirele au fost procesate, se apeleaza functia mergeAndDownloadData() pentru a unii fisierele si a le descarca in fisierul final.

```
function convertAndMergeCSV() {
  const files = document.getElementById('fileInput').files;
  let csvDataSets = [];

  function readAndParseFile(file) {
      const reader = new FileReader();
      reader.onload = function (event) {
        const csvData = event.target.result;
        const dataParsed = Papa.parse(csvData, { header: true }).data;
        csvDataSets.push(dataParsed);
        if(csvDataSets.length === files.length) {
          mergeAndDownloadData(csvDataSets);

        }

      }
      reader.readAsText(file);

    }

    for( let i = 0; i < files.length; i++ ) {
      readAndParseFile(files[i]);
    }

  }
```

5. Functia mergeDataSets() este utilizata pentru a combina datele din mai multe seturi de date reprezentate de parametrul functiei, si anume dataSets, intr-un singur set de date, prin rezolvarea eventualelor conflicte care apar intre valorile dintre fisiere, astfel:
    - Am creat un obiect gol: mergedDataSets{}. In acest obiect se vor stoca toate datele combinate. Cheia obiectului este reprezentata de “domain” (sau “root_domain” asa cum este definit in fisierul website_datasets.csv), iar valorile obiectului vor fi reprezentate de randurile associate fiecarui domeniu.
    - Se parcurg toate seturile de date, unde fiecare set de date este reprezentat de un fisier CSV, iar apoi se parcurg toate randurile aferente setului de date respectiv:
        - Pentru fiecare rand in parte se extrage cheia, care este utilizata pentru a indentifica domeniul in obiectul mergedDataSets
        - Se verifica daca a fost identificata o cheie. Daca a fost identificata cheia, se filtreaza datele din randurile: adresa, telefon, categorie si tara. De exemplu, se elimina caractere speciale aferente randurilor de pe coloana adresa sau se transforma prima litera a fiecarui cuvant din tara in majuscula.

- Apoi se verifica daca nu exista inca o intrare pentru cheie. Daca nu exista, se creeaza o noua intrare utilizand datele din randul curent.
- Daca exista deja o intrare pentru cheie, se verifica:
    - Daca valoarea existenta este goala si valoarea din randul curent este definita, atunci se va folosi valoarea din randul curent.
    - Daca valoarea existenta nu este nici goala, nici nedefinita, atunci se compara valorile din randul curent cu valorile existente si se rezolva conflictele utilizand functia resolveConflicts().

- Datele finale analizate sunt convertite apoi intr-un array de obiecte si returnate.

```
function mergeDataSets(dataSets) {
  const mergedDataSets = {};
  dataSets.forEach(files => {
    files.forEach(row => {
                const key = row['domain'] || row['root_domain'];

                if (key) {

                    let filteredAddress = row['address'] ? row['address'].replace(/[^a-zA-Z0-9 ]/g, '') : '';
                    let filteredPhone = row['phone'] ? row['phone'].replace(/[^0-9 ]/g, '') : '';
                    let filteredCategory = row['categories'] || row['category'] || row['s_category'] ? row['categories'] || row['category'] ||
                    row['s_category'].replace(/[^a-zA-Z0-9,&\-| ]/g, '') : '';
                    let filteredCountry = row['country_name'] || row['main_country'] ? (row['country_name'] ||
                     row['main_country']).toLowerCase().trim().replace(/[^a-zA-Z ]/g, '') : '';

                    filteredCountry = filteredCountry.split(' ').map(countr => countr.charAt(0).toUpperCase() + countr.slice(1)).join(' ');
                    filteredAddress = filteredAddress.split(' ').map(adr => adr.trim().replace(/[^^a-zA-Z0-9,]/g)).join(' ');

                    if (!mergedDataSets[key]) {
                        mergedDataSets[key] = { ...row, 'address': filteredAddress, 'phone' : filteredPhone, 'categories' : filteredCategory,
                        'category' : filteredCategory, 's_category' : filteredCategory, 'country_name': filteredCountry, 'main_country': filteredCountry};
                    } else {
                        Object.keys(row).forEach(field => {
                            if (mergedDataSets[key][field] === undefined || mergedDataSets[key][field] === '') {
                                mergedDataSets[key][field] = row[field];
                            } else {
                                mergedDataSets[key][field] = resolveConflicts(mergedDataSets[key][field], row[field]);
                            }
                        });
                        mergedDataSets[key]['address'] = filteredAddress;
                        mergedDataSets[key]['phone'] = filteredPhone;
                        mergedDataSets[key]['categories'] = filteredCategory;
                        mergedDataSets[key]['category'] = filteredCategory;
                        mergedDataSets[key]['s_category'] = filteredCategory;
                        mergedDataSets[key]['country_name'] = filteredCountry;
                        mergedDataSets[key]['main_country'] = filteredCountry;

                    }
                }
            });
        });
    return Object.values(mergedDataSets);

}
```

6. Functia resolveConflicts() este utilizata pentru a rezolva eventualele conflictele intre valorile existente si noile valori in timpul procesului de combinare a datelor. Aceasta functie verifica daca noua valorea ar putea sa inlocuiasca valorea existenta, prioritizand noua valoare, daca aceasta nu este nedefinita sau goala.

```
function resolveConflicts(existingValue, newValue) {
  if (newValue  !== undefined && newValue !== "") {
      return  newValue;
  }

  return existingValue;
}
```

7. Functia mergeAndDownloadData() este utilizata pentru a combina seturile de date primite ca parametru prin apelarea functiei mergeDataSets(), filtrarea datelor prin maparea ficarui rand din setul de date combinat (mergedFile) intr-un obiect care contine coloanele relevante pentru fisierul CSV, si anume: Domeniu, Categorie, Adresa, Telefon, Tara, Nume companie si descarcarea setului de date final sub forma de fisier CSV.

```
function mergeAndDownloadData(dataSets) {
  const mergedFile = mergeDataSets(dataSets);
  const filteredDataSets = mergedFile.map(row => ({
              'Domain': row['domain'] || row['root_domain'],
              'Country': row['country_name'] || row['main_country'],
              'Address': row['address'],
              'Category': row['categories'] || row['category'] || row['s_category'],
              'Phone': row['phone'],
              'Company Name': row['name'] || row['legal_name']


  }))

  const csv = Papa.unparse(filteredDataSets);
  downloadFinalCSV(csv);

}
```

8. Functia downloadFinalCSV() este utilizata pentru a descarca fisierul final compus din cele trei CSV-uri.

```
function downloadFinalCSV(csv) {
  const blob = new Blob([csv], { type: 'text/csv' });
  const urlAddress = window.URL.createObjectURL(blob);
  const el = document.createElement('a');
  el.href = urlAddress;
  el.download = 'final_merged_CSV_file.csv';
  document.body.appendChild(el);
  el.click();
  document.body.removeChild(el);
  window.URL.revokeObjectURL(urlAddress);
}
```