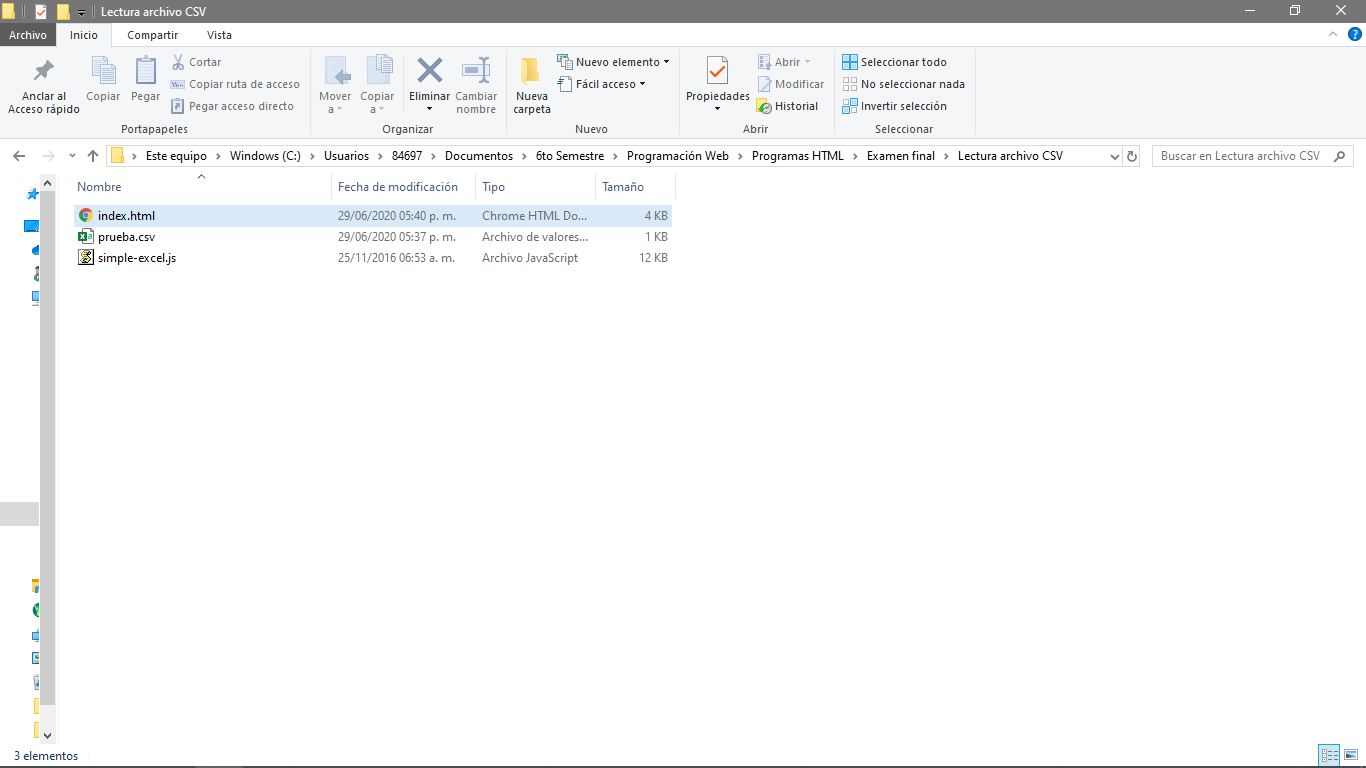
**Reporte Lectura de archivo CSV**

**Pasos para usar la aplicación:**

1.- Ejecutar el archivo index.html



2.- Arrastrar el archivo “prueba.csv” o cualquier archivo “.csv” dentro del recuadro marcado, o hacer clic en el botón “Seleccionar archivo”.

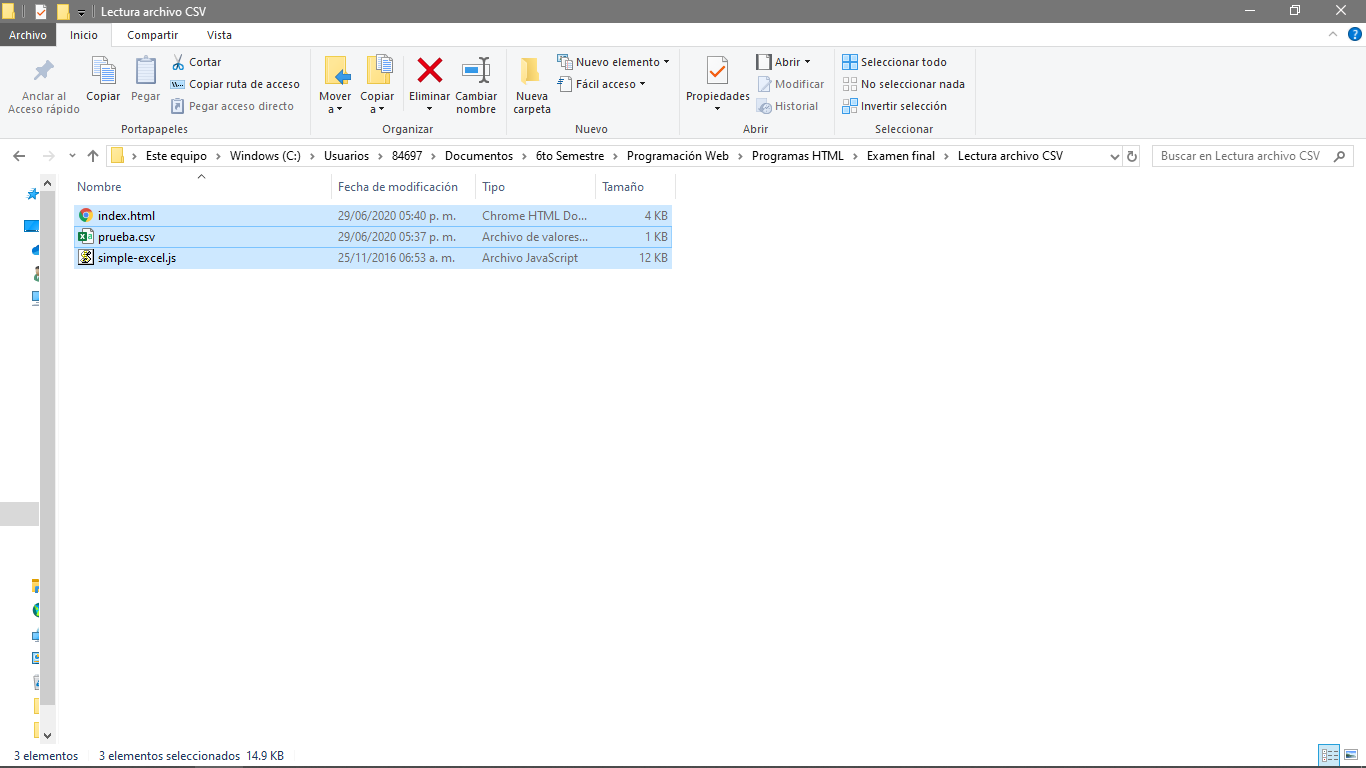
3.-Si la extensión del archivo es “.CSV“ nos aparecerá un texto de alerta donde se validó correctamente y hacemos clic en “Aceptar” para que se llenen los datos en la página web, de lo contrario tendremos que subir un archivo con la extensión “.CSV”.

4.-Observar los datos cargados del archivo “.csv” en la página web.



**Desarrollo actividad:**

Para esta actividad se ocuparon los siguientes 3 archivos:



**Index.html** es la página web donde podremos visualizar la aplicación web.

**Prueba.csv** es un archivo **“.csv”** de prueba en el cual tenemos datos guardados que posteriormente utilizaremos en la aplicación web.

**Simple-excel.js** es un archivo que contiene el script que se encuentra bajo una licencia dual de **GPLv2 & MIT** con el cual podremos leer el contenido del archivo .csv.

**Código index.html:**

<!doctype html>

<html>

    <head>

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <meta http-equiv="XA-UA-Compatible" content="ie=edge">

        <meta tittle="author" content="José Alejandro López Doblado">

        <meta title="Description" content="Práctica final Programación Web">

        <meta name="keys" content="JS, CSS, CSV, CONSOLE">

        <script type="text/javascript" src="simple-excel.js"></script>

        <style>

            \* { font-family: Helvetica, Arial, sans-serif; font-size: 12px; margin : 10px }

            td { border : 1px solid #000; padding : 10px }

            .cargar{

                border-style: dashed;

                padding: 60px;

                font-size: 17px;

                border-color: gray;

            }

            label{

                font-size: 17px;

            }

            .titulo{

                font-size: 35px;

                letter-spacing: 3px;

                padding: 30px;

                background-color: aliceblue;

            }

            .texto{

                font-style: italic;

                color: darkslategrey;

            }

            body{

                background-color: rgb(195, 233, 255);

            }

            h6{

                font-style: italic;

            }

            table, th, td{

           border: none;

            }

            td{

                background-color: aliceblue;

                text-align: center;

            }

        </style>

    </head>

    <body>

        <center>

            <br><br><br>

            <label class="titulo">Visualizar contenido archivo CSV</label><br><br><br><br><br><br>

        <label class="texto">Para cargar su archivo CSV puede hacerlo con el boton "Seleccionar archivo" o arrastrando su archivo en el recuadro marcado</label><br>

        <form method=post action="#" enctype="multipart/form-data">

            <input class="cargar" type="file" id="fileInputCSV" name="archivoupload" >

            <table id="result"></table>

            <input type=button id="fileExport" hidden="true">

            </form>

        <script type="text/javascript">

            var fileInputCSV = document.getElementById('fileInputCSV');

            // when local file loaded

            fileInputCSV.addEventListener('change', function (e) {

                var convertir = false;

                comprueba\_extension(this.form, this.form.archivoupload.value);

                //Comprueba si el archivo es .csv

                function comprueba\_extension(formulario, archivo) {

                    extensiones\_permitidas = new Array(".csv");

                    mierror = "";

                    if (!archivo) {

                        //Si no tengo archivo, es que no se ha seleccionado un archivo en el formulario

                        mierror = "No has seleccionado ningún archivo";

                    }else{

                        //recupero la extensión de este nombre de archivo

                        extension = (archivo.substring(archivo.lastIndexOf("."))).toLowerCase();

                        //alert (extension);

                        //compruebo si la extensión está entre las permitidas

                        permitida = false;

                        for (var i = 0; i < extensiones\_permitidas.length; i++) {

                            if (extensiones\_permitidas[i] == extension) {

                            permitida = true;

                            break;

                            }

                        }

                        if (!permitida) {

                            mierror = "Comprueba la extensión de los archivos a subir. \nSólo se pueden subir archivos con extensiones: " + extensiones\_permitidas.join();

                        }else{

                            //Aceptado

                            alert ("¡Todo correcto!. Se llenara la tabla con los datos.");

                            convertir=true;

                            return 0;

                        }

                    }

                    //si estoy aqui es que no se ha podido submitir

                    alert (mierror);

                    return 0;

                }

                if(convertir){

                    // parse as CSV

                    var file = e.target.files[0];

                        var csvParser = new SimpleExcel.Parser.CSV();

                        csvParser.setDelimiter(',');

                        csvParser.loadFile(file, function () {

                            // draw HTML table based on sheet data

                            var sheet = csvParser.getSheet();

                            var table = document.getElementById('result');

                            table.innerHTML = "";

                            sheet.forEach(function (el, i) {

                                var row = document.createElement('tr');

                                el.forEach(function (el, i) {

                                    var cell = document.createElement('td');

                                    cell.innerHTML = el.value;

                                    row.appendChild(cell);

                                });

                                table.appendChild(row);

                            });

                            // print to console just for quick testing

                            console.log(csvParser.getSheet(1));

                            console.log(csvParser.getSheet(1).getRow(1));

                            console.log(csvParser.getSheet(1).getColumn(2));

                            console.log(csvParser.getSheet(1).getCell(3, 1));

                            console.log(csvParser.getSheet(1).getCell(2, 3).value);

                            console.log("Creado por: José Alejandro López Doblado");

                        });

                }

            });

        </script>

        <h3>Creado por:</h3>

        <h6>

            <strong>José Alejandro López Doblado</strong>

        </h6>

    </body>

</html>

**Código prueba.csv:**

ID,Nombre,Edad,Semestre,Carrera,Instituto

1,José,23,6,Ingeniería En Sistemas Computacionales,ITCH

2,Alejandro,25,4,Ingeniería en Tecnologias de Información y Comunicación,ITCH

3,López,22,8,Licenciatura en Psicología,VISCAYA

4,Doblado,20,2,Licenciatura en Derecho,UQROO

**Código simple-excel.js:**

// SimpleExcel.js v0.1.3

// Client-side script to easily parse / convert / write any Microsoft Excel XLSX / XML / CSV / TSV / HTML / JSON / etc formats

// https://github.com/faisalman/simple-excel-js

//

// Copyright © 2013-2014 Faisal Salman <fyzlman@gmail.com>

// Dual licensed under GPLv2 & MIT

(function (window, undefined) {

    'use strict';

    ///////////////////////

    // Constants & Helpers

    ///////////////////////

    var Char = {

        COMMA           : ',',

        RETURN          : '\r',

        NEWLINE         : '\n',

        SEMICOLON       : ';',

        TAB             : '\t'

    };

    var DataType = {

        CURRENCY    : 'CURRENCY',

        DATETIME    : 'DATETIME',

        FORMULA     : 'FORMULA',

        LOGICAL     : 'LOGICAL',

        NUMBER      : 'NUMBER',

        TEXT        : 'TEXT'

    };

    var Exception = {

        CELL\_NOT\_FOUND              : 'CELL\_NOT\_FOUND',

        COLUMN\_NOT\_FOUND            : 'COLUMN\_NOT\_FOUND',

        ROW\_NOT\_FOUND               : 'ROW\_NOT\_FOUND',

        ERROR\_READING\_FILE          : 'ERROR\_READING\_FILE',

        ERROR\_WRITING\_FILE          : 'ERROR\_WRITING\_FILE',

        FILE\_NOT\_FOUND              : 'FILE\_NOT\_FOUND',

        //FILE\_EXTENSION\_MISMATCH     : 'FILE\_EXTENSION\_MISMATCH',

        FILETYPE\_NOT\_SUPPORTED      : 'FILETYPE\_NOT\_SUPPORTED',

        INVALID\_DOCUMENT\_FORMAT     : 'INVALID\_DOCUMENT\_FORMAT',

        INVALID\_DOCUMENT\_NAMESPACE  : 'INVALID\_DOCUMENT\_NAMESPACE',

        MALFORMED\_JSON              : 'MALFORMED\_JSON',

        UNIMPLEMENTED\_METHOD        : 'UNIMPLEMENTED\_METHOD',

        UNKNOWN\_ERROR               : 'UNKNOWN\_ERROR',

        UNSUPPORTED\_BROWSER         : 'UNSUPPORTED\_BROWSER'

    };

    var Format = {

        CSV     : 'csv',

        HTML    : 'html',

        JSON    : 'json',

        TSV     : 'tsv',

        XLS     : 'xls',

        XLSX    : 'xlsx',

        XML     : 'xml'

    };

    var MIMEType = {

        CSV     : 'text/csv',

        HTML    : 'text/html',

        JSON    : 'application/json',

        TSV     : 'text/tab-separated-values',

        XLS     : 'application/vnd.ms-excel',

        XLSX    : 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',

        XML     : 'text/xml',

        XML2003 : 'application/xml'

    };

    var Regex = {

        FILENAME    : /.\*\./g,

        LINEBREAK   : /\r\n?|\n/g,

        COMMA       : /(,)(?=(?:[^"]|"[^"]\*")\*$)/g,

        QUOTATION   : /(^")(.\*)("$)/g,

        TWO\_QUOTES  : /""/g

    };

    var Utils = {

        getFiletype : function (filename) {

            return filename.replace(Regex.FILENAME, '');

        },

        isEqual     : function (str1, str2, ignoreCase) {

            return ignoreCase ? str1.toLowerCase() == str2.toLowerCase() : str1 == str2;

        },

        isSupportedBrowser: function() {

            return !![].forEach && !!window.FileReader;

        },

        overrideProperties : function (old, fresh) {

            for (var i in old) {

                if (old.hasOwnProperty(i)) {

                    old[i] = fresh.hasOwnProperty(i) ? fresh[i] : old[i];

                }

            }

            return old;

        }

    };

    /////////////////////////////

    // Spreadsheet Constructors

    ////////////////////////////

    var Cell = function (value, dataType) {

        var defaults = {

            value    : value || '',

            dataType : dataType || DataType.TEXT

        };

        if (typeof value == typeof {}) {

            defaults = Utils.overrideProperties(defaults, value);

        }

        this.value = defaults.value;

        this.dataType = defaults.dataType;

        this.toString = function () {

            return value.toString();

        };

    };

    var Records = function() {};

    Records.prototype = [];

    Records.prototype.getCell = function(colNum, rowNum) {

        return this[rowNum - 1][colNum - 1];

    };

    Records.prototype.getColumn = function (colNum) {

        var col = [];

        this.forEach(function (el, i) {

            col.push(el[colNum - 1]);

        });

        return col;

    };

    Records.prototype.getRow = function (rowNum) {

        return this[rowNum - 1];

    };

    var Sheet = function () {

        this.records = new Records();

    };

    Sheet.prototype.getCell = function (colNum, rowNum) {

        return this.records.getCell(colNum, rowNum);

    };

    Sheet.prototype.getColumn = function (colNum) {

        return this.records.getColumn(colNum);

    };

    Sheet.prototype.getRow = function (rowNum) {

        return this.records.getRow(rowNum);

    };

    Sheet.prototype.insertRecord = function (array) {

        this.records.push(array);

        return this;

    };

    Sheet.prototype.removeRecord = function (index) {

        this.records.splice(index - 1, 1);

        return this;

    };

    Sheet.prototype.setRecords = function (records) {

        this.records = records;

        return this;

    };

    /////////////

    // Parsers

    ////////////

    // Base Class

    var BaseParser = function () {};

    BaseParser.prototype = {

        \_filetype   : '',

        \_sheet      : [],

        getSheet    : function(number) {

            number = number || 1;

            return this.\_sheet[number - 1].records;

        },

        loadFile    : function (file, callback) {

            var self = this;

            //var filetype = Utils.getFiletype(file.name);

            //if (Utils.isEqual(filetype, self.\_filetype, true)) {

                var reader = new FileReader();

                reader.onload = function () {

                    self.loadString(this.result, 0);

                    callback.apply(self);

                };

                reader.readAsText(file);

            //} else {

                //throw Exception.FILE\_EXTENSION\_MISMATCH;

            //}

            return self;

        },

        loadString  : function (string, sheetnum) {

            throw Exception.UNIMPLEMENTED\_METHOD;

        }

    };

    // CSV

    var CSVParser = function () {};

    CSVParser.prototype = new BaseParser();

    CSVParser.prototype.\_delimiter = Char.COMMA;

    CSVParser.prototype.\_filetype = Format.CSV;

    CSVParser.prototype.loadString = function (str, sheetnum) {

        // TODO: implement real CSV parser

        var self = this;

        sheetnum = sheetnum || 0;

        self.\_sheet[sheetnum] = new Sheet();

        str.replace(Regex.LINEBREAK, Char.NEWLINE)

           .split(Char.NEWLINE)

           .forEach(function(el, i)

        {

            var sp = el.split(Regex.COMMA);

            var row = [];

            sp.forEach(function(cellText) {

                if (cellText !== self.\_delimiter) {

                    cellText = cellText.replace(Regex.QUOTATION, "$2");

                    cellText = cellText.replace(Regex.TWO\_QUOTES, "\"");

                    row.push(new Cell(cellText));

                }

            });

            self.\_sheet[sheetnum].insertRecord(row);

        });

        return self;

    };

    CSVParser.prototype.setDelimiter = function (separator) {

        this.\_delimiter = separator;

        return this;

    };

    // HTML

    var HTMLParser = function () {};

    HTMLParser.prototype = new BaseParser();

    HTMLParser.prototype.\_filetype = Format.HTML;

    HTMLParser.prototype.loadString = function(str, sheetnum) {

        var self = this;

        var domParser = new DOMParser();

        var domTree = domParser.parseFromString(str, MIMEType.HTML);

        var sheets = domTree.getElementsByTagName('table');

        sheetnum = sheetnum || 0;

        [].forEach.call(sheets, function(el, i) {

            self.\_sheet[sheetnum] = new Sheet();

            var rows = el.getElementsByTagName('tr');

            [].forEach.call(rows, function (el, i) {

                var cells = el.getElementsByTagName('td');

                var row = [];

                [].forEach.call(cells, function (el, i) {

                    row.push(new Cell(el.innerHTML));

                });

                self.\_sheet[sheetnum].insertRecord(row);

            });

            sheetnum++;

        });

        return self;

    };

    // TSV

    var TSVParser = function () {};

    TSVParser.prototype = new CSVParser();

    TSVParser.prototype.\_delimiter = Char.TAB;

    TSVParser.prototype.\_filetype = Format.TSV;

    // XML

    var XMLParser = function () {};

    XMLParser.prototype = new BaseParser();

    XMLParser.prototype.\_filetype = Format.XML;

    XMLParser.prototype.loadString = function(str, sheetnum) {

        var self = this;

        var domParser = new DOMParser();

        var domTree = domParser.parseFromString(str, MIMEType.XML);

        var sheets = domTree.getElementsByTagName('Worksheet');

        sheetnum = sheetnum || 0;

        [].forEach.call(sheets, function(el, i) {

            self.\_sheet[sheetnum] = new Sheet();

            var rows = el.getElementsByTagName('Row');

            [].forEach.call(rows, function (el, i) {

                var cells = el.getElementsByTagName('Data');

                var row = [];

                [].forEach.call(cells, function (el, i) {

                    row.push(new Cell(el.innerHTML));

                });

                self.\_sheet[sheetnum].insertRecord(row);

            });

            sheetnum++;

        });

        return self;

    };

    // Export var

    var Parser = {

        CSV : CSVParser,

        HTML: HTMLParser,

        TSV : TSVParser,

        XML : XMLParser

    };

    /////////////

    // Writers

    ////////////

    // Base Class

    var BaseWriter = function () {};

    BaseWriter.prototype = {

        \_filetype   : '',

        \_mimetype   : '',

        \_sheet      : [],

        getSheet    : function(number) {

            number = number || 1;

            return this.\_sheet[number - 1].records;

        },

        getString   : function () {

            throw Exception.UNIMPLEMENTED\_METHOD;

        },

        insertSheet : function (data) {

            if (!!data.records) {

                this.\_sheet.push(data);

            } else {

                var sheet = new Sheet();

                sheet.setRecords(data);

                this.\_sheet.push(sheet);

            }

            return this;

        },

        removeSheet : function (index) {

            this.\_sheet.splice(index - 1, 1);

            return this;

        },

        saveFile    : function () {

            // TODO: find a reliable way to save as local file

            window.open('data:' + this.\_mimetype + ';base64,' + window.btoa(this.getString()));

            return this;

        }

    };

    // CSV

    var CSVWriter = function () {};

    CSVWriter.prototype = new BaseWriter();

    CSVWriter.prototype.\_delimiter = Char.COMMA;

    CSVWriter.prototype.\_filetype = Format.CSV;

    CSVWriter.prototype.\_mimetype = MIMEType.CSV;

    CSVWriter.prototype.getString = function () {

        // TODO: implement real CSV writer

        var self = this;

        var string = '';

        this.getSheet(1).forEach(function (el, i) {

            el.forEach(function (el) {

                string += el + self.\_delimiter;

            });

            string += '\r\n';

        });

        return string;

    };

    CSVWriter.prototype.setDelimiter = function (separator) {

        this.\_delimiter = separator;

        return this;

    };

    // TSV

    var TSVWriter = function () {};

    TSVWriter.prototype = new CSVWriter();

    TSVWriter.prototype.\_delimiter = Char.TAB;

    TSVWriter.prototype.\_filetype = Format.TSV;

    TSVWriter.prototype.\_mimetype = MIMEType.TSV;

    // Export var

    var Writer = {

        CSV : CSVWriter,

        TSV : TSVWriter

    };

    /////////////

    // Exports

    ////////////

    var SimpleExcel = {

        Cell                : Cell,

        DataType            : DataType,

        Exception           : Exception,

        isSupportedBrowser  : Utils.isSupportedBrowser(),

        Parser              : Parser,

        Sheet               : Sheet,

        Writer              : Writer

    };

    window.SimpleExcel = SimpleExcel;

})(this);