Universidad de San Carlos de Guatemala
Facultad de ingeniería
Ingeniería en Ciencias y Sistemas
Introducción a la programación y computación 1

MANUAL TECNICO

Integrantes:

Alberto Josue Hernández Armas 201903553
Kevin Mark Hernández Chicol 202001053
Leonel Antonio González García 201709088
Tulio Jafeth Pirir Schuman 201700698

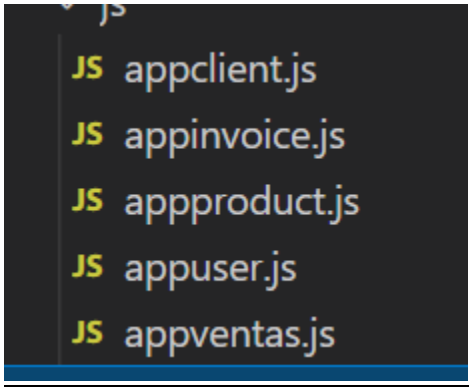
Guatemala06 de julio de 2021

INTRODUCCION

En el presente manual técnico podremos observar cómo es que está diseñado el código y explicando que forman la funcionalidad de una manera general, teniendo como objetivo principal que sea más entendible. Determinando los métodos utilizados y explicando algunas palabras claves refiriéndonos a sus propiedades demostrando cuál es su función dentro de los bloques de código que se nos presentan a continuación.

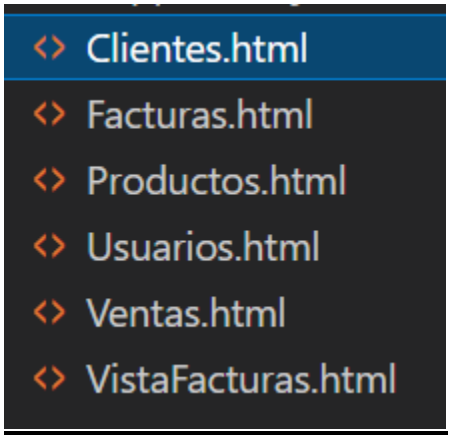
Manual Técnico

Un archivo *. js, es un archivo de texto plano que contiene scripts de Javascript, y que puede, por tanto, ser modificado con cualquier editor de textos. Es ejecutado generalmente por un navegador web.



Un archivo con la extensión de archivo HTM o HTML es un archivo de Lenguaje de marcado de hipertexto y es el tipo de archivo de página web estándar en Internet.

Los archivos HTM y HTML también pueden hacer referencia a otros archivos como videos, CSS o archivos JS.



Métodos y Funciones

```
!DOCTYPE html>
html lang="en">

head>
 <meta charset="UTF-8">
 <title>Reportes</title>
 <link rel="stylesheet" href="https://maxcdn.b  Follow link (ctrl + click)  trap/4.2.1/css/bootstrap.mi
 <link rel="stylesheet" type="text/css" href="css/estilo.css">
 <meta name="viewport"
   content="width=device-width, user-scalable=no, initial-scale=1.0, maximun-scale=1.0, minimu
/head>
```

Inicio y creación de un archivo html , ,hterf = css/estilos. csshace referencia a un archivo css está diseñada para marcar la separación del contenido de las páginas web y la forma de presentación de estas.

Bootstrap es un framework popular gratuito de HTML, CSS y JavaScript que se usa para desarrollar sitios web interactivos principalmente para dispositivos móviles.

```
body>
 <div class="contenedor">
   <header id="hd">
     <input id="inputJson" type="file" />
     <img src="img/reporte.png" id="logo">
   </header>
   <header id="hhd">
     <ul class="menu">
       <li class="boton"><a href="Usuarios.html">Usuarios</a></li>
       <li class="boton"><a href="Clientes.html">Clientes</a></li>
       <li class="boton"><a href="Productos.html">Productos</a></li>
       <li class="boton"><a href="Facturas.html">Facturas</a></li>
       <li class="boton"><a href="FacturasRango.html">Facturas-Rango</a></li>
       <li class="boton"><a href="Ventas.html">Ventas</a></li>
       <li class="boton"><a href="Top5.html">Top5</a></li>
     </ul>
   </header>
   <section class="main">
     <div id = "TodosClientes" class="d-flex justify-content-around flex-wrap">
     </div>

     <div id="exampleModal" class="modal fade" tabindex="-1">
       <div id = "VistaIn" class="modal-dialog modal-sm">

       </div>
     </div>
     <script src="js/appclient.js"></script>
   </section>
   <footer>
     <p>
     <h5>Grupo 16 &copy;</h5>
     </p>
   </footer>

 </div>
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" in
/body>
```

El cuerpo de los html contiene etiquetas de diferentes tipos como botones divisiones, footers etc. Así como referencias a otros archivos html.

```
  </div>
  <script src="js/appclient.js"></script>
</section>
```

Script crc hace referencia a un archivo javaScriopt donde estarán todas las funciones que se podrán ejecutar desde el html, sin esta ninguna funcionalidad seri posible.

```
function leerJSON() {
    const fileReader = new FileReader()

    function miOnLoad() {
        const json = JSON.parse(fileReader.result)
        localStorage.setItem("clientesS", JSON.stringify(json)
        const f = JSON.parse(localStorage.getItem("clientesS")
        crearTarjetas(f)
    }

    fileReader.readAsText(inputJson.files[0])
    fileReader.onload = miOnLoad
}
```

Esta función lee un archivo json y lo guarda en el localStoradge . localStorage y sessionStorage son propiedades que acceden al objeto Storage y tienen la función de almacenar datos de manera local, la diferencia entre éstas dos es que localStorage almacena la información de forma indefinida o hasta que se decida limpiar los datos del navegador y sessionStorage almacena información mientras la pestaña donde se esté utilizando siga abierta, una vez cerrada, la información se elimina.

```
function crearTarjeta(cliente) {
    let html = `
        <div class="card mb-4 text-white bg-primary" style="width: 18rem;">
        <div class="card-body">
        <h5 class="card-title">${cliente.id}</h5>
        </div>
        <ul "class="list-group list-group-flush">
        <li class="list-group-item list-group-item-info">Nombre: ${cliente.name}</li>
        <li class="list-group-item list-group-item-info">Direccion: ${cliente.address}</li>
        <li class="list-group-item list-group-item-info">Tel: ${cliente.phone}</li>
        <li class="list-group-item list-group-item-info">NIT: ${cliente.nit}</li>
        </ul>
        <div class="card-body">
        <a><button type="button" class="btn-ver btn btn-warning"  data-bs-toggle="modal" data
        </div>
        </div>`
    return html
}
```

Función crea una tarjeta con los datos obtenidos de un archivo Json..

```
function BuscarCliente(id, clientes) {
    for (const cliente of clientes) {
        if (cliente.id == id) {
            ht = Ver(cliente.id, cliente.name, cliente.address, cliente.phone, cliente.nit)
             Vista.innerHTML= ht;
        }
    }
}
```

Busca en el Storeg y muestra la información requerida

```
const on = (element, event, selector, handler) => {
    console.log(handler)
    element.addEventListener(event, e => {
        if (e.target.closest(selector)) {
            handler(e)
        }
    })
}

if (location.reload) {
    const f = JSON.parse(localStorage.getItem("clientesS"))
    if (f != null) {
        crearTarjetas(f)
        //localStorage.removeItem("clientesS",JSON.stringify(f))
    }
}


inputJson.addEventListener('change', leerJSON)

on(document, 'mouseover', '.btn-ver', e => {
    const tarjeta = e.target.parentNode.parentNode.parentNode
    const divt = tarjeta.firstElementChild
    const id = divt.firstElementChild.innerHTML
    const clientes = JSON.parse(localStorage.getItem("clientesS"))
    BuscarCliente(id, clientes)

})
```

Manejo de LocalStoradge para el almacenamiento de los datos json

```
localStorage.setItem("montoordenado",JSON.stringify(cajita))
localStorage.setItem("idordenado",JSON.stringify(cajita2))
localStorage.setItem("nombreordenado",JSON.stringify(cajita3))

}
function nombresordenados(clientes)
{
    for (const p of cajita2) {
        for (const j of clientes) {
            if (p == j.id) {cajita3.push(j.name)}
        }
    }

}
```

Referencia al local storadge y como se va a guardar

```
function NuevaIndi(invoices, clientes, tot, productos){
    var I ={
        id: invoices.id,
        clienteid: clientes.id,
        clientenombre: clientes.name,
        clientedireccion:clientes.address,
        clientephone: clientes.phone,
        clientenit: clientes.nit,
        fecha: invoices.date,
        total: tot,
        products: [productos]
    };
    IndiF.push(I)
    localStorage.setItem('infiF', JSON.stringify(IndiF));
    IndiF.pop(I)
}

const on = (element, event, selector, handler) => {
    console.log(handler)
    element.addEventListener(event, e => {
        if (e.target.closest(selector)) {
            handler(e)
        }
    })
}
```

Función Busca por id al los Clientes

```
function AlmacenarFactura(id, facturas, clientes) {
    var tot=0;
    for (var i = 0; i < facturas.length; i++) {
       if(id==facturas[i].id){
        for (var j = 0; j < clientes.length; j++) {
            if (facturas[i].client == clientes[j].id) {
                for(var k=0; k<facturas[i].products.length; k++){
                    tot += facturas[i].products[k].price
                }
                NuevaIndi(facturas[i], clientes[j], tot, facturas[i].products)
                tot = 0;
            }
        }
       }
    }
}
```

Guarda y almacena las facturas.

```
inputJson.addEventListener('change', leerJSON)

on(document, 'mouseover', '.btn-ver', e => {
    const tarjeta = e.target.parentNode.parentNode.parentNode
    const divt = tarjeta.firstElementChild
    const id = divt.firstElementChild.innerHTML
    const productos = JSON.parse(localStorage.getItem("prroducto"))
    BuscarProducto(id, productos)

})
```

Input para la carga del archivo json.

```
var fechas = [];
var G_clients=[];
function FacturasRangosFechas(){
    //Rango de Fechas
    // obtener fechas del imput
    var RangoFechas=[];
    var d= document.getElementById("d").value;
    var m= document.getElementById("m").value;
    var a= document.getElementById("a").value;

    var df= document.getElementById("df").value;
    var mf= document.getElementById("mf").value;
    var af= document.getElementById("af").value;
    console.log(d,"-",m,"-",a)
    console.log(df,"-",mf,"-",af)



    //Pasar el imput a tipo Fecha

    var fecha1=(a+'-'+m+'-'+d);
    console.log(fecha1)

    var fecha2=(af+'-'+mf+'-'+df);
    var fe1=new Date(fecha1);
    fe1.setHours(0,0,0,0);
    var fe2=new Date(fecha2);
    fe2.setHours(0,0,0,0);


    //Se llena con las fehcas de json un var Rango de Fechas para compararlo
     for (let i = 0; i < fechas.length; i++) {
        var f1 = new Date(fechas[i].date);
        f1.setHours(0,0,0,0);
        RangoFechas.push(f1)
    }
    //Ordenamiento de Rango de Fechas de mayor a menor
    var n, i, k, aux;
    n = RangoFechas.length;
    // Mostramos, por consola, la RangoFechas desordenada
    // Algoritmo de burbuja
    for (k = 1; k < n; k++) {
        for (i = 0; i < (n - k); i++) {
            if (RangoFechas[i].getTime() > RangoFechas[i + 1].getTime()) {
                aux = RangoFechas[i];
                RangoFechas[i]= RangoFechas[i + 1];
                RangoFechas[i + 1] = aux;
            }
        }
    }
    //var conparar para comparar si esta en el rango o no
    var conparar=[];
    for (let x = 0; x < RangoFechas.length; x++) {

        if ((RangoFechas[x].getTime()>=fe1.getTime())&&(RangoFechas[x].getTime()<=fe2.getTime())) {
            conparar[x]=RangoFechas[x].toDateString()


        }
    }
```

Guarda almacena y muestra las facturas de a cuerdo a las fechas especificadas

```
// Algoritmo de burbuja
for (k = 1; k < n; k++) {
    for (i = 0; i < (n - k); i++) {
        if (RangoFechas[i].getTime() > RangoFechas[i + 1].getTime()) {
            aux = RangoFechas[i];
            RangoFechas[i]= RangoFechas[i + 1];
            RangoFechas[i + 1] = aux;
        }
    }
}
```

Ordena y muestra los datos.