

Documentación

abril 2

2018

Yasiel
Hernández.

Contenido

Contenido	2
Introducción incidencias trello.....	3
Tecnologías usadas para trello.....	3
Obtener Claves de acceso:	3
El código:	3
Vista de la página.	5
Introducción agenda.	7
Ficheros usador:	7
Tecnologías usadas para la agenda.....	7
Recibir los datos de los empleados.	7
Generar agenda en formato de cartas.....	9
Filtrado de la agenda.....	10
Filtrado por el nombre:	11
Vista principal de la agenda:	11
Introducción reservas de sala.	12
Ficheros usador:	12
Tecnologías usadas para la agenda.....	12
Consultar la base de datos.	12
Crear Botones.....	13
Crear Calendario:	14
Generar fechas y reservas visualmente del calendario	14
Modal de reservas.....	17
Insertar una reserva:	18
Eliminar una reserva.....	20
Instalar apache:	21
Instalar php 7.2.....	21
Configurar conexión a SQL Server:.....	21
Instalar MYSQL	21
Instalar y configurar phpMyAdmin	22
Consulta php para los datos a SQL Express desde Linux:.....	22

INCIDENCIAS TRELLO

Introducción incidencias trello.

El proyecto consiste en generar tarjetas en trello de forma automatizada, para que cualquier usuario con problemas pueda reportar una incidencia y de forma automática le llegue la notificación a los encargados de resolver dicha incidencia.

Tecnologías usadas para trello.

HTML, CSS, JavaScript, Bootstrap y la aplicación de trello.

Obtener Claves de acceso:

La “key” que te facilita la página de desarrolladores de trello: <https://developers.trello.com/>

El “token” que se obtiene una vez registrado y obtenido la “key”: <https://trello.com/app-key>

Un “idlist” necesario para saber donde insertarás la tarjeta, el “idlist” se encuentra en la ruta del proyecto con la terminación json:



El código:

Para poder generar la incidencia de una manera correcta debemos tener en cuenta el navegador desde el cual se crea la incidencia debido a que hay navegadores que no permiten obtener la ip del equipo, por eso condicionamos la creación de la incidencia:

```
//-----EN CASO DE QUE SE USE INTERNET EXPLORER O EDJE-----
var es_ie = navigator.userAgent.indexOf("MSIE") > -1;
if (getBrowserInfo() == 'IE 11' || getBrowserInfo() == 'Edge 16') {
    let descripcion = document.getElementById("incidenciasDescripcion").value;
    var desc = descripcion + '%0A' + " Ip no disponible al usar Internet Explore. " + "Sistema operativo: " + OSName;
    crearCarta(desc, OSName, fechaTrello);
} else { //-----DEMAS NAVEGADORES-----
```

Ya verificado el navegador procedemos a crear la carta, pasándole la descripción el sistema operativo y la fecha en el formato exigido. En la creación de la carta tenemos que tener en cuenta que para poder insertar documentos adjuntos, labels y miembros tenemos que recoger **la id de la carta**.

```
xhr.addEventListener("readystatechange", function () {
    if (this.readyState === this.DONE) {
        console.log(this.responseText);
        var dt = this.responseText;
        h = JSON.parse(dt).id;
    }
});
```

Ya obtenido el id de la carta, ya podemos llamar a las demás funciones:

1. Tenemos todas las claves almacenadas en variables:

```
var appkey = "151bcd104f1742fdc0b8c2f4a4c8764";
var secret = "c5a52ad53cef30fb0539bab09df6967178a40d187ef829ae9c93faf700ea6d16";

var token = "ddc55434f6f11fbc1a3379adde4d5f66cd8be4be97d4d90eaca39322af045925";
var idlist = "5aaf6422caeb39da694e7dc1";
var usuario1 = "5891c93eb1cfa471ee1fe47c";
var usuario2 = "59a68c4e314350c790512ae9";
```

- Primero comprobamos si es importante la incidencia, en caso de que este marcado se le asignará un color a la tarjeta, para añadir ese color específico es necesario añadir el id de ese color de esa manera:

```
if (importante.checked == true) {  
    var checkRQ = new XMLHttpRequest();  
    checkRQ.open("POST", "https://api.trello.com/1/cards/" + data + "/idLabels?value=5aaf6396841642c2a8277156&key=" + appkey +  
    checkRQ.send(datas);  
}
```

- Para adjuntar documentos se ha creado un array el cual si está vacío no se ejecuta la función, y directamente pasará a insertar los usuarios predefinidos. En el proceso adjuntar los archivos a la tarjeta trello tenemos que tener en cuenta que no se cierre la conexión hasta que se suban todos los archivos.

```
request.addEventListener("readystatechange", function () {  
    if (this.readyState === this.DONE) {  
        var finalizado = true;  
        for (let i = 0; i < arrData.length; i++) {  
            if (arrData[i].readyState !== this.DONE) {  
                finalizado = false;  
            }  
        }  
        if (finalizado == true) {  
            spinner.style = 'display:none';  
            $("#mensajeModal").modal();  
            var close = document.getElementById('close');  
            close.addEventListener('click', function () {  
                location.reload();  
            }, false)  
        }  
    }  
});
```

- Para insertar los miembros debemos obtener la id de dicho miembro, **para obtener la id de dicho miembro**, debemos agregar de manera manual los miembros desde trello y buscar su id en el json igual que hicimos con el idlist.

```
{ "id": "59a68c4e314350c790512ae9", "avatarHash": "2b0e1ebe5c5c245db533430d893dcd3", "fullName": "Borja", "initials": "B", "username": "borja383" }
```

```
function usuarioPredefinido(data) {  
    var arrRQ = [];  
    var datas = null;  
    var usuRQ1 = new XMLHttpRequest();  
    usuRQ1.open("POST", "https://api.trello.com/1/cards/" + data + "/idMembers?value=" + usuario1 + "&key=" + appkey + "&token=" + token);  
    usuRQ1.send(datas);  
    var usuRQ2 = new XMLHttpRequest();  
    usuRQ2.open("POST", "https://api.trello.com/1/cards/" + data + "/idMembers?value=" + usuario2 + "&key=" + appkey + "&token=" + token);  
    usuRQ2.send(datas);  
}
```

Incidencias

Utilice el siguiente formulario para crear una incidencia.

Nombre (*)

Juan

Título (*)

No funciona la navegacion de la pagina principal

Descripción

No carga el sistema

Haz click aquí si es urgente

Seleccionar archivo

Los campos con asterisco son requeridos.

Limpiar

Crear

Estructura base de datos MYSQL (phpMyAdmin)

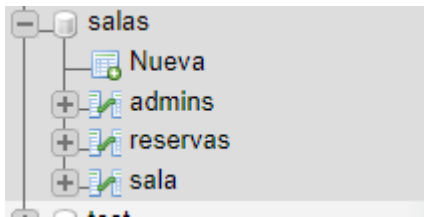


Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
<input type="checkbox"/> admins	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16 KB	-
<input type="checkbox"/> reservas	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	latin1_swedish_ci	16 KB	-
<input type="checkbox"/> sala	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	latin1_swedish_ci	16 KB	-
3 tablas	Número de filas	9	InnoDB	latin1_swedish_ci	48 KB	0 B

admins:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	usuario	varchar(100)	latin1_swedish_ci		No	Ninguna			Cambiar Eliminar Primaria Único Índice Espacial ▼ Más
<input type="checkbox"/> 2	password	varchar(100)	latin1_swedish_ci		No	Ninguna			Cambiar Eliminar Primaria Único Índice Espacial ▼ Más

reservas:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	ID	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Primaria Único Índice ▼ Más
<input type="checkbox"/> 2	entrada	datetime			No	Ninguna			Cambiar Eliminar Primaria Único Índice ▼ Más
<input type="checkbox"/> 3	salida	datetime			No	Ninguna			Cambiar Eliminar Primaria Único Índice ▼ Más
<input type="checkbox"/> 4	sala	int(11)			No	Ninguna			Cambiar Eliminar Primaria Único Índice ▼ Más
<input type="checkbox"/> 5	Usuario	int(11)			No	Ninguna			Cambiar Eliminar Primaria Único Índice ▼ Más
<input type="checkbox"/> 6	motivo	varchar(100)	latin1_swedish_ci		No	Ninguna			Cambiar Eliminar Primaria Único Índice ▼ Más

Sala:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	ID	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Primaria Único Índice ▼ Más
<input type="checkbox"/> 2	nombre	varchar(100)	latin1_swedish_ci		No	Ninguna			Cambiar Eliminar Primaria Único Índice ▼ Más

Agenda

Introducción agenda.

La agenda consiste en tener algunos de los datos de los empleados de la empresa de manera organizada, bajo un filtro en el cual puedas buscar por la compañía o el nombre, la agenda se genera bajo una consulta a la base de datos.

Ficheros usador:

Index.html, cartas.js, cartas.php

Tecnologías usadas para la agenda.

HTML, CSS, JavaScript, PHP, Bootstrap, MYSQL.

Recibir los datos de los empleados.

Para recibir los datos de los empleados lo primero que hacemos es una **consulta PHP (llamada desde el código JavaScript)** a la base de datos donde está alojada de esta manera:

```
<?php
//-----Consulta Agenda-----
$pdo=new PDO("sqlsrv:Server=172.26.7.192;Database=A3LABORAL", "consulta", "Monte00!");
$stmtement=$pdo->prepare("select * from [master].[dbo].[ZMontesano_Vista_Agenda]");
$stmtement->execute();
if (!$stmtement){
    echo 'Error al ejecutar la consulta';
}else{
    $results = $stmtement->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($results);
}

?>
```

Esos datos que se reciben en la consulta convertidos previamente en formato JSON los almacenamos en un array del código, para luego trabajar directamente sobre ese array.

La llamada al php y como almacenamos los datos:

```
var registroCC = [];
if (window.XMLHttpRequest) { ...
} else { ...
}
xmlhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
        var fechas = JSON.parse(this.responseText);
        console.log(fechas);
        for (let i = 0; i < fechas.length; i++) {
            registroCC.push(fechas[i]);
        }

        var contenedordatalistagenda = document.getElementById("personasagenda");
        var contenedordatalistincidencias = document.getElementById("personasincidencias");

        registroCC.forEach(e => { ...
        });

        registroCC.forEach(e => { ...
        });

        var filteredEmpresas = [];

        registroCC.forEach(element => { ...
        });

        var contenedorempresas = document.getElementById('empresas');

        filteredEmpresas.forEach(e => { ...
        });
    }
};
xmlhttp.open("GET", "php/cartas.php", true);
console.log(xmlhttp);
xmlhttp.send();
```

Para poder tener un filtrado autocompletado metemos los datos recibidos de manera organizada:

```
var contenedordatalistagenda = document.getElementById("personasagenda");
var contenedordatalistincidencias = document.getElementById("personasincidencias");

registroCC.forEach(e => {
    var name = e.CompleteName;

    var option = document.createElement("option");
    option.value = name;
    option.innerText = name;

    contenedordatalistagenda.appendChild(option);
});

registroCC.forEach(e => {
    var name = e.CompleteName;

    var option = document.createElement("option");
    option.value = name;
    option.innerText = name;

    contenedordatalistincidencias.appendChild(option);
});

var filteredEmpresas = [];

registroCC.forEach(element => {
    if (filteredEmpresas.indexOf(element.CompanyName) == -1) {
        filteredEmpresas.push(element.CompanyName);
    }
});
```



```

var conentadorempresas = document.getElementById('empresas');

filteredEmpresas.forEach(e => {
  var option = document.createElement("option");
  option.value = e;
  option.innerText = e;

  conentadorempresas.appendChild(option);
});
});
xmlhttp.open("GET", "php/cartas.php", true);
console.log(xmlhttp)
xmlhttp.send();

```

Con esto conseguimos rellenar los inputs según los datos recibidos de esta manera:

<div> <div>Empresa</div> <div>TODAS</div> <div> <div>TODAS</div> <div>LOGISTICA MONTESANO SL</div> <div>MONTESANO CANARIAS SA</div> <div>GARCIA PASCUAL CANARIAS SA</div> <div>MONTESANO EXTREMADURA SA</div> <div>DIACA CINCO SA</div> <div>DIACA DOS SA</div> <div>DIACA SL</div> <div>AGROPECUARIA MONTESANO SL</div> </div> </div>	<div> <div>Nombre a buscar</div> <div>Juan</div> <div> <div>SOSA ALMEIDA, JUAN MANUEL</div> <div>GARCIA LORENZO, JUAN</div> <div>HERRERA MARTIN, JUAN MANUEL</div> <div>VAZQUEZ MALDONADO, JUAN</div> <div>REYES VELAZQUEZ, JUAN MIGUEL</div> <div>DE LA CRUZ SANCHEZ, JUAN J</div> <div>CALDERON ROQUE, JUAN JESUS</div> <div>TORRES BACALLADO, JUAN CARLOS</div> <div>ESTEVEZ DUREY, JUAN MIGUEL</div> <div>HERNANDEZ PLASENCIA, JUAN JOSE</div> <div>CORTES FERNANDEZ, JUAN ALBERTO</div> <div>SANCHEZ LASSO, JUAN ANTONIO</div> <div>GARCIA MESA, JUAN JOSE</div> <div>VAZQUEZ TRIGO, JUAN EUSEBIO</div> <div>RODRIGUEZ HERNANDEZ, JUAN CARLOS</div> </div> </div>
--	--

Generar agenda en formato de cartas.

Para generar todas cartas tenemos sin filtrado tenemos la función generarCartas() que recorre el array con todos los datos y llama a la función carta() que crea de manera visual la información:

```

var contAgenda = document.getElementById('contagenda');
var buscador = document.getElementById('buscador');

function generarCartas() {
  pill.style = 'display:flex;';
  contAgenda.innerHTML = '';
  for (let i = 0; i < registroCC.length; i++) {
    carta(i);
  }
}

```

```

function carta(i) {
    var card = document.createElement('div');
    card.setAttribute('class', 'card text-black border-dark bg-info mb-3 carta');
    card.setAttribute('style', 'max-width: 23rem; margin:10px;');
    card.setAttribute('id', 'carta' + i);
    var header = document.createElement('div');
    header.setAttribute('class', 'card-header');
    var cardBody = document.createElement('div');
    cardBody.setAttribute('class', 'card-body');
    cardBody.setAttribute('id', 'contenidoCarta');
    var titulo = document.createElement('h5');
    titulo.setAttribute('class', 'card-title');

    var h = registroCC[i].Observations;
    var exten = '';
    if(h.indexOf('#') === -1){
        var ca = h.split('#')[1];
        exten = ca.split('#')[0];
    }

    var texto = document.createElement('div');
    header.innerHTML = '<b>Persona</b>';
    titulo.innerHTML = '<b>' + registroCC[i].Name + ' ' + registroCC[i].SecondName1 + ' ' + registroCC[i].SecondName2 + '</b>';
    texto.innerHTML = '<b> Lugar de trabajo </b>' + registroCC[i].WorkplaceName + '<br> <b>Fijo: </b>' + registroCC[i].FixedPhone;

    cardBody.appendChild(titulo);
    cardBody.appendChild(texto);
    card.appendChild(header);
    card.appendChild(cardBody);
    contAgenda.appendChild(card);
}

```

Filtrado de la agenda.

Para filtrar **por compañía** tenemos la función selEmpresa() la cual se ejecuta cada vez que seleccionamos una empresa esa función se encarga de pasrle los valores seleccionados a getDom() la función la cual **condiciona la manera en que se debe llamar a carta()** .

```

function getDom(valSelect) {
    var palabra = buscador.value.toUpperCase();
    empresa = valSelect;
    if (empresa === 'TODAS' && palabra === '') {
        generarCartas();
    }
    for (let i = 0; i < registroCC.length; i++) {
        if (empresa === registroCC[i].CompanyName && palabra === (registroCC[i].Name + ' ' + registroCC[i].SecondName1 + ' ' + registroCC[i].SecondName2)) {
            carta(i);
        } else if (empresa === registroCC[i].CompanyName && palabra === (registroCC[i].Name + ' ' + registroCC[i].SecondName1)) {
            carta(i);
        } else if (empresa === registroCC[i].CompanyName && palabra === (registroCC[i].SecondName1 + ' ' + registroCC[i].SecondName2)) {
            carta(i);
        } else if (empresa === registroCC[i].CompanyName && palabra === (registroCC[i].Name + ' ' + registroCC[i].SecondName1)) {
            carta(i);
        } else if (empresa === registroCC[i].CompanyName && palabra === (registroCC[i].SecondName1 + ' ' + registroCC[i].SecondName2)) {
            carta(i);
        } else if (empresa === registroCC[i].CompanyName && palabra === '') {
            carta(i);
        } else if (empresa === registroCC[i].CompanyName && palabra === registroCC[i].Name) {
            carta(i);
        } else if (empresa === 'TODAS' && palabra === (registroCC[i].Name + ' ' + registroCC[i].SecondName1)) {
            carta(i);
        } else if (empresa === 'TODAS' && palabra === (registroCC[i].Name + ' ' + registroCC[i].SecondName2)) {
            carta(i);
        } else if (empresa === 'TODAS' && palabra === (registroCC[i].SecondName1 + ' ' + registroCC[i].Name)) {
            carta(i);
        } else if (empresa === 'TODAS' && palabra === (registroCC[i].Name + ' ' + registroCC[i].SecondName1 + ' ' + registroCC[i].SecondName2)) {
            carta(i);
        } else if (empresa === 'TODAS' && palabra === '') {
            carta(i);
        } else if (empresa === 'TODAS' && palabra === registroCC[i].Name) {
            carta(i);
        }
    }
}

```

Filtrado por el nombre:

Este tipo de filtrado se produce al escribir un nombre en el input que lo solicita y seguidamente darle a enter. Para realizar este filtrado llamamos a la función search() la cual genera la carta según el nombre y la empresa que esté seleccionada en ese momento.

```
function search() {
    contAgenda.innerHTML = '';
    var palabra = buscador.value.toUpperCase();
    var selIndex = document.getElementById("empresas").selectedIndex;
    var selValue = document.getElementById("empresas").options[selIndex].innerHTML;

    if (palabra == "") {
        generarCartas();
    } else {
        for (let i = 0; i < registroCC.length; i++) {
            if (registroCC[i].CompleteName.indexOf(palabra) >= 0) {
                carta(i);
            } else if (palabra == (registroCC[i].Name + ' ' + registroCC[i].SecondName1)) {
                if (selValue == registroCC[i].CompanyName) {
                    carta(i);
                } else if (selValue == 'TODAS') {
                    carta(i);
                }
            } else if (palabra == (registroCC[i].Name + ' ' + registroCC[i].SecondName2)) {
                if (selValue == registroCC[i].CompanyName) {
                    carta(i);
                } else if (selValue == 'TODAS') {
                    carta(i);
                }
            } else if (palabra == (registroCC[i].SecondName1 + ' ' + registroCC[i].Name)) {
                if (selValue == registroCC[i].CompanyName) {
                    carta(i);
                } else if (selValue == 'TODAS') {
                    carta(i);
                }
            } else if (palabra == (registroCC[i].Name + ' ' + registroCC[i].SecondName1 + ' ' + registroCC[i].SecondName2)) {
                if (selValue == registroCC[i].CompanyName) {
                    carta(i);
                } else if (selValue == 'TODAS') {
                    carta(i);
                }
            } else if (palabra == (registroCC[i].SecondName1 + ' ' + registroCC[i].SecondName2 + ' ' + registroCC[i].Name)) {
                if (selValue == registroCC[i].CompanyName) {
                    carta(i);
                } else if (selValue == 'TODAS') {
                    carta(i);
                }
            }
        }
    }
}
```

Vista principal de la agenda:

Montesano Incidentes Agenda Reserva de salas

Empresa TODAS

Nombre a buscar

Persona

JUAN MANUEL SOSA ALMEIDA

Lugar de trabajo LOGISTICA MONTESANO
SL (LAS PALMAS)
Fijo: Extensión de fijo 0
Móvil Extensión:
Número de Fax
Observaciones
Código Completo 10120030

Persona

JOSE DAVID GONZALEZ GONZALEZ

Lugar de trabajo MONTESANO CANARIAS
SA
Fijo: Extensión de fijo 0
Móvil Extensión:
Número de Fax
Observaciones
Código Completo 10010782

Persona

RAUL GARCIA PASCUAL

Lugar de trabajo GARCIA PASCUAL
CANARIAS SA
Fijo: Extensión de fijo 0
Móvil Extensión:
Número de Fax
Observaciones
Código Completo 10119905

Persona

RAUL GARCIA PASCUAL

Lugar de trabajo MONTESANO
EXTREMADURA SA
Fijo: Extensión de fijo 0
Móvil Extensión:
Número de Fax
Observaciones
Código Completo 10068865

Reservas de salas

Introducción reservas de sala.

La reservas de sala consisten en tener un calendario en el cual puedas realizar una reserva de una sala concreta, visualizar las reservas ya creadas, incluso eliminar una reserva si se tiene permiso, todo ello consultando una base de datos.

Ficheros usador:

Index.html, crearClendarios.js, datosCalendar.js, modalCalendar.js, admin.js, reservas.php, insertar.php, salas.php, delete.php, admin.php

Tecnologías usadas para la agenda.

HTML, CSS, JavaScript, PHP, Bootstrap, MYSQL.

Consultar la base de datos.

Para recibir todos los datos tenemos una consulta php que es llamada desde recogerDatos() almacenando todos esos datos en un array en el mismo código, la consulta php devuelve un JSON.

Consulta PHP (reservas.php)

```
<?php
//-----Select de las reservas-----
$pdo=new PDO("mysql:dbname=salas;host=127.0.0.1","root","");
$stmt=$pdo->prepare("SELECT * FROM reservas");
$stmt->execute();
if (!$stmt){
    echo 'Error al ejecutar la consulta';
}else{
    $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($results);
}

?>
```

Llamada a la consulta php donde recibimos todos los datos (crearCalendario.js)

```
var registro = [];

function recogerDatos(idSala) {
    if (window.XMLHttpRequest) {
        xmlhttp = new XMLHttpRequest();
    } else {
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            var fechas = JSON.parse(this.responseText);
            console.log(fechas);
            for (let i = 0; i < fechas.length; i++) {
                registro.push(fechas[i]);
            }
            if(idSala == undefined){
            }else{
                document.getElementById('sala'+idSala).click(idSala);
            }
        }
    };
    xmlhttp.open("GET", "php/reservas.php", true);
    console.log(xmlhttp)
    xmlhttp.send();
}
```

Crear Botones

Para crear el botón que luego generará un calendario según la sala que sea tenemos la **función crearBtns()** que recorre la base de datos distinguiendo cuantas salas hay y le añade al botón el nombre de la sala. Para ello tenemos una llamada **php a la tabla de salas** en la base de datos. Recogiendo los datos en un array en el código JavaScript, de esta manera:

(salas.php)

```
<?php
//-----Select de las reservas-----
$pdo=new PDO("mysql:dbname=salas;host=127.0.0.1","root","");
$stmt=$pdo->prepare("SELECT * FROM sala");
$stmt->execute();
if (!$stmt){
    echo 'Error al ejecutar la consulta';
}else{
    $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($results);
}

?>
```

(CC.js)

```
var salas = []
recogerSalas()

function recogerSalas() {
    if (window.XMLHttpRequest) {
        xmlhttp = new XMLHttpRequest();
    } else {
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            var sala = JSON.parse(this.responseText);

            for (let i = 0; i < sala.length; i++) {
                salas.push(sala[i]);
            }
            console.log(salas);
            crearBtns();
        }
    };
    xmlhttp.open("GET", "php/salas.php", true);
    console.log(xmlhttp)
    xmlhttp.send();
}
```

Crear Calendario:

Para crear el calendario lo primero es distinguir la sala en la cual se genera el calendario. Para ello le pasamos un identificador mediante el botón al hacer click.

(crearCalendario.js)

```
function crearBtns() {  
  var Arbtn = document.getElementsByClassName('salas');  
  const unique = [...new Set(registro.map(item => item.sala))];  
  var contenedorbotones = document.getElementById("contenedorbotones");  
  for (let i = 1; i <= salas.length; i++) {  
  
    var cbtn = document.createElement('button')  
    cbtn.setAttribute('type', 'button');  
    cbtn.setAttribute('class', 'btn btn-outline-primary salas btn-salas');  
    cbtn.setAttribute('id', 'sala' + i);  
    cbtn.setAttribute('onclick', 'crearCalendarioGlobal("' + i + '")');  
    cbtn.innerHTML = 'Salas ' + salas[i - 1].nombre  
    contenedorbotones.appendChild(cbtn);  
  }  
}
```

Identificador:

1,2,3....

Al pulsar cualquier botón llamamos a la función crearCalendarioGlobal() la cual se encarga de llamar a crearCalendario() pasándole el identificador, crearCalendario() dibuja crea todos los elementos del calendario, y luego llama a currDate().

(crearCalendario.js)

```
function crearCalendarioGlobal(e) {  
  creaCalendario(e);  
  document.getElementById("selectedSala").value = e;  
  var titulo = document.getElementById('tituloSala');  
}
```

Generar fechas y reservas visualmente del calendario

Pasos para generar un aspecto como este:



En el paso anterior después de crear los ítems del calendario llamábamos a `currDate()` pasándole el identificador, `currDate` se encarga de recoger el mes y el año actual y añadirse los dos ítems del `index.html`, procedemos a la parte final de la creación del calendario **llamando a `datosCalendar()` pasándole el identificador de sala**

(`datosCalendar.js`)

```
function currDate(e) {
    var currDate = new Date();
    document.getElementById('mes').value = currDate.getMonth() - 1;

    document.getElementById('anio').value = currDate.getFullYear();
    datosCalendar(e);
}
```

Procedemos a rellenar los ítems del calendario según el mes en el que estemos. En esta hay varias partes a tener en cuenta:

(`datosCalendar.js`)

```
function datosCalendar(e) {
    var days = document.querySelectorAll(".day");
    days.forEach(element => {
        element.style = 'background-color: none;';
    });
    var insertado = false;
    var n;
    if (e == "+1" || e == "-1") {
        n = parseInt(e);
    } else {
        n = parseInt("1");
    }

    var month = document.getElementById('mes').value;
    var year = document.getElementById('anio').value;
    if (isNaN(n)) {} else if (n == 1) {
        month++;
        document.getElementById('mes').value++;
        if (month > 11) {
            document.getElementById('anio').value++;
            document.getElementById('mes').value = 1
        }
    } else {
        month--;
        document.getElementById('mes').value--;
        if (month == -1) {
            document.getElementById('anio').value--;
            document.getElementById('mes').value = 12
        }
    }
}
```

Esta función llama de dos formas diferentes, o bien dándole al botón del cual procedemos o al hacer clic en el mes siguiente dentro del mismo calendario.

En el caso que proceda del calendario `datosCalendar` recibe "+1" o "-1" según si le a dado al mes siguiente o anterior.

Lo primero es parsear dichos elementos y luego procedemos a restar o sumar un mes al mes mostrado en el calendario.

Luego procedemos a dibujar todos los elementos del calendario

(`datosCalendar.js`)

```

for (var i = 0, j = 1, dd = 1, da = difference; i < divDays.length; i++) {
    if (i >= FirstdayOfWeek - 1)
        if (j <= daysInMonth) {
            divDays[i].innerHTML = j;
            var dc = String(j)
            divDays[i].setAttribute('id', '' + j + ' ' + (month + 1));
            if (dc.length == 1) {
                divDays[i].setAttribute('id', '0' + j + ' ' + (month + 1))
            }
            divDays[i].setAttribute('onclick', 'abrirmodal(this.id)');
            j++;
        } else {
            divDays[i].innerHTML = '<o>' + dd + '</o>';
            var db = String(dd)
            divDays[i].setAttribute('class', 'day siguientes');
            divDays[i].setAttribute('id', '' + dd + ' ' + (month + 2));
            if (db.length == 1) {
                divDays[i].setAttribute('id', '0' + dd + ' ' + (month + 2));
            }
            divDays[i].setAttribute('onclick', 'abrirmodal(this.id)');
            dd++;
        }
    } else {
        divDays[i].innerHTML = '<o>' + da + '</o>';
        var de = String(da)
        divDays[i].setAttribute('class', 'day pasados');
        divDays[i].setAttribute('id', '' + da + ' ' + month);
        if (de.length == 1) {
            divDays[i].setAttribute('id', '0' + da + ' ' + month)
        }
        divDays[i].setAttribute('onclick', 'abrirmodal(this.id)');
        da++;
    }
}

```

Este bucle se encarga de distinguir los días mostrados del mes pasado, los del mes actual y los del mes siguiente, añadiéndole a su vez un id con dicho el día y el mes, también le añadimos un onclick en el cual al hacer clic dentro del recuadro de ese día abrirá un modal pasándole la id.

(datosCalendar.js)

```

for (let j = 0; j < registro.length; j++) {
    if (registro[j].sala == e) {
        var diasplit = registro[j].entrada.substring(10, 8);
        if (auxdia != diasplit) {
            texto = "";
        }
        var messplit = registro[j].entrada.substring(5, 7);
        var newDiasSplit = parseInt(diasplit, 10);
        var newMesSplit = parseInt(messplit, 10);
        var diaLi = diasplit + ' ' + newMesSplit;
        var coderight = registro[j].Usuario.substring(4, 8);
        var codeleft = registro[j].Usuario.substring(0, 4);
        var employeecode = "00" + coderight;
        var employeeName = "";
        registroCC.forEach(e => {
            if (e.EmployeeCode == employeecode) {
                employeeName = e.Name;
            }
        });
        texto += '' + employeeName + ' <br>' + registro[j].entrada.substring(11, 16) + ' - ' + re
        insertado = true;
        try {
            document.getElementById(diaLi).setAttribute('data-original-title', texto);
            var m = document.getElementById(diaLi).getAttribute('data-original-title');
            //----- INSERTAR COLOR AL CUADRADO DEL CALENDARIO-----
            if (m.length > 0) {
                document.getElementById(diaLi).style = 'background-color: rgba(199, 31, 31, 0.411)';
            } else {
                document.getElementById(diaLi).style = 'background-color: none;';
            }
        } catch (error) {
        }
        auxdia = registro[j].entrada.substring(10, 8);
    }
}

```

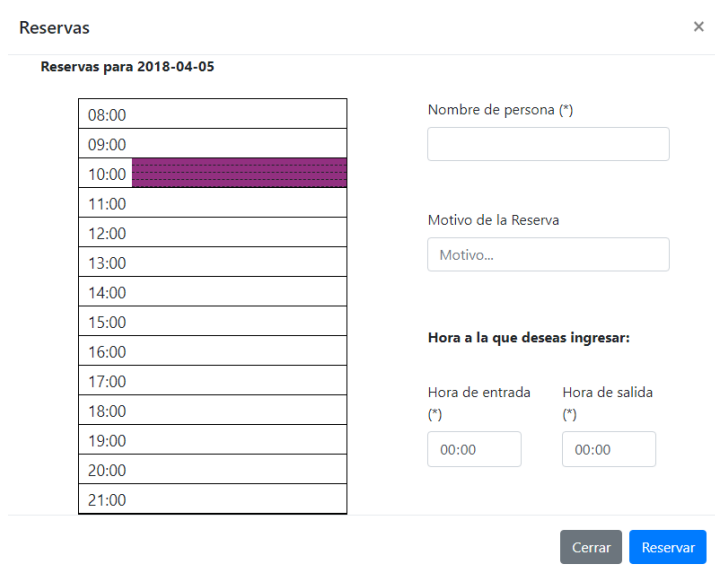
Este bucle se encarga de comprobar en el array de las reservas (registro) todas las reservas hechas marcándolas de otro color, y añadiéndole un tooltip que al pasar el ratón por encima muestra, la información de la reserva, y la persona que hizo dicha reserva recogiendo su código de la base de datos almacenadas en el array "registro".

Modal de reservas.

Al ser clic en cualquiera de los días se abrirá un modal cargando la función **abrirmodal()** (en “modalCalendar.js”) al cual se le pasa el id del día clicado.

Lo primero que se hace al ejecutar esta función es crearEsquema() (en “crearCalendario.js”) la cual crea los ítems que en función de las horas libres o reservadas rellenaremos desde abrirmodal().

Esta función genera este aspecto visual:



El esquema de las horas las rellenamos comprobando en el array “registro” que horas están ocupadas.

Por cada registro pintamos las horas seleccionadas y le añadimos: un tooltip con nombre y hora de la reserva y un onclick que cargará la función de eliminar el registro (bajo usuario y contraseña).

```
for (let i = 0; i < registro.length; i++) {
  if (registro[i].entrada.substring(10, -1) == seleccionado && registro[i].sala == selectedSala) {
    var randomrgba = random_rgba();

    var coderight = registro[i].Usuario.substring(4, 8)
    var codeleft = registro[i].Usuario.substring(0, 4);
    var employeecode = "00" + coderight;

    var employeeName = "";
    registroCC.forEach(e => {
      if (e.EmployeeCode == employeecode) {
        employeeName = e.Name;
      }
    });
  }
};
```

Ejemplo de una condición para rellenar el esquema de las horas.

```
});

for (let t = 0; t < hh.length; t++) {

  if (hh[t].id == registro[i].entrada.substring(11, 16)) {
    hh[t].style = 'background-color:' + randomrgba + ';margin: 0; height:8px;width:80%;position:rel
    hh[t].setAttribute('data-toggle', "modal");
    hh[t].setAttribute('data-html', "true");
    hh[t].setAttribute('data-placement', "top");
    hh[t].setAttribute('data-original-title', employeeName+ '<br><p>Concepto: '+registro[i].motivo+
    hh[t].setAttribute('data-target', '#ModalAdmin');
    hh[t].setAttribute('onclick', 'eliminarReserva(this)')
    hh[t].setAttribute('reserva', registro[i].ID);
  }
  if (hh[t].id > registro[i].entrada.substring(11, 16) && hh[t].id < registro[i].salida.substring(11,
```

Insertar una reserva:

```
$("#btn-ingresar").click(function () {  
  
    //-----AÑADIMOS NUEVA FECHA -----  
    var horaEntrada = document.getElementById('horaEntrada').value;  
    var horaSalida = document.getElementById('horaSalida').value;  
    var fechaEntrada = seleccionado + ' ' + horaEntrada + ':00';  
    var fechaSalida = seleccionado + ' ' + horaSalida + ':00';  
    var idUsuario = document.getElementById('usuarioElegido').value;  
    var motivo = document.getElementById('motivoReserva').value;  
    var idSala = document.getElementById('selectedSala').value;  
    var us = ''  
    var v = true;  
    for (let i = 0; i < registro.length; i++) {  
        if (fechaEntrada > fechaSalida || fechaEntrada == fechaSalida) {  
            alert('Error salida antes de la entrada');  
            v = false;  
            break;  
        }  
        if (new Date(fechaEntrada) >= new Date(registro[i].entrada) && new Date(fechaSalida) <= new Date(registro[i].salida)) {  
            alert('Error esa hora está ocupada <b>');  
            v = false;  
            break;  
        }  
        if (new Date(fechaEntrada) <= new Date(registro[i].entrada) && new Date(fechaSalida) >= new Date(registro[i].salida)) {  
            alert('Error esa hora está ocupada');  
            v = false;  
            break;  
        }  
        if (new Date(fechaEntrada) <= new Date(registro[i].entrada) && new Date(fechaSalida) <= new Date(registro[i].salida)) {  
            alert('Error esa hora está ocupada');  
            v = false;  
            break;  
        }  
    }  
}
```

Al rellenar todos los datos de la reserva y darle al botón de reservar se ejecuta una función la cual comprueba que esas horas no estén reservadas en primer

```
for (let i = 0; i < registroCC.length; i++) {  
    if(registroCC[i].CompleteName == idUsuario){  
        us = registroCC[i].CompanyCode + registroCC[i].EmployeeCode.substring(2);  
    }  
}  
if (idUsuario == '' || idUsuario == null) {  
    alert('Error hay que seleccionar un usuario');  
    v = false;  
}  
if (motivo == '' || motivo == null){  
    alert('Añada un motivo')  
    v = false;  
}
```

Comprobamos que todos los campos para la reserva estén rellenos.

(modalCalendar.js)

```
if (v == true) {  
    var infoParaEnviar = {  
        sala: idSala,  
        usuario: us,  
        entrada: fechaEntrada,  
        salida: fechaSalida,  
        motivo: motivo  
    };  
    $.ajax({  
        type: "POST",  
        url: "php/insertar.php",  
        data: infoParaEnviar,  
        dataType: "text",  
        async: false,  
        success: function () {  
            recogerDatos(idSala)  
        }  
    });  
};
```

Para insertar los datos en la base de datos con la nueva reserva hacemos una llamada AJAX haciendo un POST a insertar.php.

Al cumplirse esa condición vuelve a mostrar el calendario.

(insertar.php)

```
#!/usr/bin/php  
//-----Insertar una reserva-----  
$sala = $_POST['sala'];  
$usuario = $_POST['usuario'];  
$fechaEntrada = $_POST['entrada'];  
$fechaSalida = $_POST['salida'];  
$motivo = $_POST['motivo'];  
  
$ins = new PDO("mysql:dbname=salas;host=127.0.0.1", "root", "");  
$insert = $ins->prepare("INSERT INTO `reservas` (`entrada`, `salida`, `sala`, `Usuario`, `motivo`) VALUES (CAST('" . $fechaEntrada . "' AS DATETIME),  
$insert->execute();  
?<
```

```
$("#calendarioModal").modal("hide");  
  
$("#modalreservas").modal();  
  
var name = '';  
var fullcode = us;  
  
var coderight = fullcode.substring(4, 8);  
  
var employeeecode = "00" + coderight;  
  
registroCC.forEach(e => {  
    if (e.EmployeeCode == employeeecode) {  
        name = e.CompleteName;  
    }  
});  
  
var mensaje = "Reserva confirmada para <b>" + name + "</b> desde las <b>" +  
document.getElementById("reservaInformacion").innerHTML = mensaje;  
var close = document.getElementById('close');
```

Finalmente lanzamos un mensaje con la reserva creada y los datos de la reserva.

Reserva confirmada

Reserva creada.

Reserva confirmada para **RODRIGUEZ REYES, JESUS** desde las **08:00** hasta las **09:00** del día **05-04-2018**

Close

Eliminar una reserva

Al hacer clic dentro de la reserva n el modal, se **ejecutará la función eliminarReservasa()** (en CC.js)

Primero se cargará un modal el cual solicitará un usuario y contraseña.



The image shows a modal window titled "Solo administradores" with a close button (X) in the top right corner. Inside the modal, there are two input fields: "Usuario" and "Password". At the bottom of the modal, there are two buttons: "Close" (grey) and "Save changes" (blue).

(CC.js)

```
if (admin[i].usuario == usu && admin[i].password == pass) {  
  
    var infoParaEnviar = {  
        reserva: rr  
    };  
    $.ajax({  
        type: "POST",  
        url: "php/delete.php",  
        data: infoParaEnviar,  
        dataType: "text",  
        async: false,  
        success: function () {  
            }  
        });  
    $('#ModalAdmin').modal('hide');  
    $('#calendarioModal').modal('hide');  
    $('#mensajeModal').modal();  
  
    var titleC = document.getElementById('titleC');  
    titleC.innerHTML = 'Reserva eliminada correctamente </br>';  
  
    var close = document.getElementById('close');  
    close.addEventListener('click',function () {  
        window.location.reload();  
    })  
} else {  
    alert('Usuario o contraseña incorrectos!')  
}
```

En caso de que el usuario y la contraseña estén en la base de datos haremos una llamada AJAX al insertar.php enviándole los datos para proceder con la eliminación de la reserva en la base de datos.

Luego mostramos un mensaje confirmando la eliminación de la reserva

(eliminar.php)

```
<?php  
//-----Eliminar una reserva-----  
$id = $_POST['reserva'];  
  
$del = new PDO("mysql:dbname=salas;host=127.0.0.1", "root", "");  
$delete = $del->prepare("DELETE FROM reservas WHERE ID=$id ");  
$delete->execute();  
?>
```

Instalación en el Servidor (Apache, PHP 7.2, Conexión SQL Server, MYSQL, PhpMyAdmin)

Los pasos para instalar nuestra web en el servidor son los siguientes:

Instalar apache:

Desde la terminal ejecutamos el siguiente comando:

```
Sudo apt-get install apache2
```

Instalar php 7.2

Desde la terminal ejecutamos los siguientes comandos:

```
add-apt-repository ppa:ondrej/php -y
apt-get update
apt-get install php7.2 php7.2-dev php7.2-xml -y --allow-unauthenticated
```

Configurar conexión a SQL Server:

Según la versión de Linux seguimos los pasos de este enlace:

<https://docs.microsoft.com/es-es/sql/connect/odbc/linux-mac/installing-the-microsoft-odbc-driver-for-sql-server?view=sql-server-2017>

Seguidamente ejecutamos estos comandos:

```
echo extension=pdo_sqlsrv.so >> `php --ini | grep "Scan for additional .ini files" |
sed -e "s|.*/\s*||"/30-pdo_sqlsrv.ini
echo extension=sqlsrv.so >> `php --ini | grep "Scan for additional .ini files" | sed -e
"s|.*/\s*||"/20-sqlsrv.ini
exit
sudo pecl install sqlsrv
sudo pecl install pdo_sqlsrv
```

```
apt-get install libapache2-mod-php7.2 apache2
a2dismod mpm_event
a2enmod mpm_prefork
a2enmod php7.2
echo "extension=sqlsrv.so" >> /etc/php/7.2/apache2/php.ini
echo "extension=pdo_sqlsrv.so" >> /etc/php/7.2/apache2/php.ini
```

Instalar MYSQL

Desde la terminal ejecutamos el siguiente comando:

```
Sudo apt-get install mysql-server
```

Instalar y configurar phpMyAdmin

Descargar la última versión de:

<https://www.phpmyadmin.net/downloads/>

Extraer en:

```
/usr/share/phpmyadmin
```

Dentro de /usr/share/phpmyadmin modificamos:

```
define('CONFIG_DIR',''); → cambiamos por: define('CONFIG_DIR', '/etc/phpmyadmin/')
```

Consulta php para los datos a SQL Express desde Linux:

```
<?php
$serverName = "172.26.7.192";
$connectionOptions = array(
    "Database" => "A3LABORAL",
    "Uid" => "consulta",
    "PWD" => "Monte00!"
);
//Establishes the connection
$arr = array();
$conn = sqlsrv_connect($serverName, $connectionOptions);
if($conn){
    $tsql= "select * from [master].[dbo].[ZMontesano_Vista_Agenda]";
    $getResults= sqlsrv_query($conn, $tsql);
    if ($getResults == FALSE){
        die(FormatErrors(sqlsrv_errors()));
    }else{
        while ($row = sqlsrv_fetch_array($getResults, SQLSRV_FETCH_ASSOC)) {
            array_push($arr, $row);
        }
        sqlsrv_free_stmt($getResults);
    }
    echo json_encode($arr);
}
?>
```