**MTC Established cmp:**

<aura:component controller="MTC\_EstablishedController" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Attribute declaration-->

<aura:attribute name="establishlist" type="MTCs\_Established\_\_c[]"/>

<aura:attribute name="respValue" type="String" />

<aura:attribute name="rowCount" type="Integer" default="0"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!-- <aura:attribute name="validatecode" type="boolean" default="false"/>-->

<aura:attribute name="showModal" type="boolean" default="false"/>

<lightning:notificationsLibrary aura:id="notifyId"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!--Action method/doInit-->

<aura:handler name="init" value="{!this}" action="{!c.addRow}"/>

<aura:html tag="style">

<!--.toastMessage.forceActionsText {

font-weight: bolder !important;

} -->

</aura:html>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 70rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario de establecimiento de Centro</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

<!--lightning components for designing UI-->

<div class="slds-m-around--x--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!-- <img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Times New Roman; color:green;">Formulario de establecimiento de Centro </span>

</div>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-cell-wrap" >S.No</div>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="ID" align="center" >ID<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<div class="slds-cell-wrap" title="State Code" align="center">Código del Estado<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Location Code" align="center">Código de Localización<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Location Name" align="center">Nombre del lugar<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-cell-wrap" title="Date of Establishment" align="center">Fecha de Fundación<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<div class="slds-cell-wrap" title="Name of MTC" align="center">Nombre de MTC<abbr class="slds-required" >\*</abbr>

</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Date operational" align="center">Fecha de apertura<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="Date closed" align="center">Fecha de cierre<abbr class="slds-required" >\*</abbr></div>

</th>-->

<!-- <th scope="col">

<div class="slds-truncate" title="Action" align="center">Action</div>

</th>-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.establishlist}" var="est" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<!-- <td>

<lightning:input name="ID" type="text" value="{!Far.ID\_\_c}" />

</td>-->

<td>

<!-- <div class="{!v.validatecode=='false' ? 'Validate' : 'message slds-form-element' }">-->

<lightning:input name="State Code" type="text" minlength="2" maxlength="2" value="{!est.State\_Code\_\_c}" variant="label-hidden" pattern="[0-9]\*" class="Stateno"/>

</td>

<td>

<lightning:input name="Location Code" type="text" minlength="2" maxlength="2" value="{!est.Location\_Code\_\_c}" variant="label-hidden" pattern="[0-9]\*" class="Stateno" />

</td>

<td>

<lightning:input name="Location Name" type="text" value="{!est.Location\_Name\_\_c}" variant="label-hidden" />

</td>

<!-- <td>

<lightning:input type="text" name="input" pattern="^((0[1-9])|([12][0-9])|(3[01]))\/((0[1-9])|(1[0-2]))\/((19[0-9]{2})|(2[0-9]{3}))$" required="true" value="{!est.Date\_of\_Establishment\_\_c}" class="date" placeholder="DD/MM/AAAA" variant="label-hidden"/>

</td>-->

<td>

<lightning:input name="Name of MTC" type="text" value="{!est.Name\_of\_MTC\_\_c}" variant="label-hidden" />

</td>

<td>

<!-- <lightning:input name="Date operational" type="date" value="{!est.Date\_Operational\_\_c}"/>-->

<lightning:input type="text" name="input" pattern="^((0[1-9])|([12][0-9])|(3[01]))\/((0[1-9])|(1[0-2]))\/((19[0-9]{2})|(2[0-9]{3}))$" required="true" value="{!est.Date\_Operational\_\_c}" class="date" placeholder="DD/MM/AAAA" variant="label-hidden"/>

</td>

<!-- <td>

<lightning:input name="Date closed" type="date" value="{!est.Date\_Closed\_\_c}" />

</td> -->

<!-- <td>

<a onclick="{!c.removeRow}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td> -->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

<!--New box code -->

<aura:if isTrue="{!v.showModal}">

<section role="dialog" tabindex="-1" aria-labelledby="modal-heading-01"

aria-modal="true" aria-describedby="modal-content-id-1"

class="slds-modal slds-fade-in-open">

<div class="slds-modal\_\_container">

<header class="slds-modal\_\_header">

<lightning:buttonIcon iconName="utility:close"

onclick="{! c.hideModel }"

alternativeText="close"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h2 id="modal-heading-01" class="slds-text-heading\_medium slds-hyphenate">

Modal Box</h2>

</header>

<!--Modal/Popup Box Body Starts here-->

<div class="slds-modal\_\_content slds-p-around\_medium" id="modal-content-id-1">

<p><b>Modal Popup Box Example Bharath: {!v.respValue}

</b>

</p>

</div>

<!--Modal/Popup Box Footer Starts here-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="neutral"

label="Cancel"

title="Cancel"

onclick="{! c.hideModel }"/>

<lightning:button variant="brand"

label="OK"

title="OK"

onclick="{!c.saveDetails}"/>

</footer>

</div>

</section>

<div class="slds-backdrop slds-backdrop\_open"></div>

</aura:if>

</aura:component>

**Controller:**

({

//Select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.establishlist");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//add new row method

addRow: function(component, event, helper) {

helper.addMTCRecord(component, event);

},

//remove row method(single row)

removeRow: function(component, event, helper) {

//Get the Farmer list

var establishlist = component.get("v.establishlist");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

establishlist.splice(index, 1);

component.set("v.establishlist", establishlist);

},

//final method to create record in server side

save: function(component, event, helper) {

if (helper.validateMTCRecords(component, event)) {

helper.saveMTCList(component, event, helper);

}

//helper.showModel(component, event, helper);

//component.set("v.showModal", true);

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

// Showing the values in Popup while uploading files in both .csv and .xlsx format

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("CodigodelEstado", "CodigodeLocalizacion", "Nombredellugar", "NombredeMTC" , "Fechadeapertura");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

//alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

// alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

// alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("CodigodelEstado", "CodigodeLocalizacion", "Nombredellugar", "NombredeMTC" , "Fechadeapertura");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

// alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

// Validating and Saving the records while uploading both .csv and .xlsx format

readCSV: function(component, event, helper) {

const headers2 = new Array("CodigodelEstado", "CodigodeLocalizacion", "Nombredellugar", "NombredeMTC" , "Fechadeapertura");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

// alert('rows>>' + rows);

var headers = rows[0].split(',');

// alert('headers>>' + headers);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

//alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].CodigodelEstado=="" ||

data1[k].CodigodeLocalizacion==""||

data1[k].Nombredellugar=="" ||

data1[k].NombredeMTC=="" ||

data1[k].Fechadeapertura==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var CodigodelEstado;

var CodigodeLocalizacion;

var Nombredellugar;

var NombredeMTC;

var Fechadeapertura;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

//alert(rows);

arr.pop();

// alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

//alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].CodigodelEstado=="" ||

data1[k].CodigodeLocalizacion==""||

data1[k].Nombredellugar=="" ||

data1[k].NombredeMTC=="" ||

data1[k].Fechadeapertura==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var CodigodelEstado;

var CodigodeLocalizacion;

var Nombredellugar;

var NombredeMTC;

var Fechadeapertura;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

addMTCRecord: function(component, event) {

//get the MTC List from component

var establishlist = component.get("v.establishlist");

//alert(JSON.stringify(establishlist));

//console.log('establishlist@@'+JSON.stringify(establishlist));

//Add New MTC Record

establishlist.push({

'sobjectType': 'MTCs\_Established\_\_c',

'State\_Code\_\_c':'',

'Location\_Code\_\_c':'',

'Location\_Name\_\_c':'',

// 'Date\_of\_Establishment\_\_c':'',

'Name\_of\_MTC\_\_c':'',

'Date\_Operational\_\_c': '',

'ConcatValue': '',

// 'Date\_Closed\_\_c':'',

});

component.set("v.establishlist", establishlist);

},

//checking validation for all required fields

validateMTCRecords: function(component, event) {

//Validate all account records

var isValid = true;

var establishlist = component.get("v.establishlist");

console.log('establishlist>>'+ establishlist);

for (var i = 0; i < establishlist.length; i++) {

if (establishlist[i].State\_Code\_\_c == '' ||

establishlist[i].Location\_Code\_\_c == '' ||

establishlist[i].Location\_Name\_\_c == '' ||

establishlist[i].Date\_of\_Establishment\_\_c =='' ||

establishlist[i].Name\_of\_MTC\_\_c == '' ||

establishlist[i].Date\_Operational\_\_c == '') {

// establishlist[i].Date\_Closed\_\_c == '') {

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " +(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

//Saving the records while uploading files

CreateContact : function (component,jsonstr){

console.log('jsonstr' + jsonstr);

//alert('inside createcontact');

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

//commented below by Dhivya

/\* action.setCallback(this, function(response) {

var state = response.getState();

//alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

alert('result callback>>'+ result);

alert("Records Inserted Succesfully");

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

}); \*/

action.setCallback(this,function(response){

//get the response state

var state = response.getState();

//alert(state);

// alert('values ');

var res = response.getReturnValue().isSuccessName;

// alert('res'+res);

component.set("v.respValue", res);

//component.set("v.showModal", true);

var rlt = component.get("v.respValue");

//document.getElementById("rlt").style.color = "#ff0000";

// alert(rlt);

console.log('@@@@result'+rlt);

//check if result is successfull

if(state == "SUCCESS"){

//Reset Form

// alert('state success');

console.log(response.getReturnValue());

if(!response.getReturnValue().isError){

if(response.getReturnValue().isDuplicate){

// alert('MTC Record is duplicate with following records'+ response.getReturnValue().isDuplicateName);

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message":"Su MTC ya está creado, continúe con el formulario de Participante : "+ response.getReturnValue().isDuplicateName,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "sticky"

});

toastEvent.fire();

}else{

//alert('Record Created Succesfully'+ response.getReturnValue().isSuccessName);

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "éxito!",

"message": "Registro MTC guardado con éxito y tu código MTC es : "+ response.getReturnValue().isSuccessName ,

duration: '20000',

key: "info\_alt",

type: "success",

mode: "sticky"

});

toastEvent.fire();

//window.location.href='/s/thank-you-page';

//window.location.reload();

$A.get('e.force:refreshView').fire();

}

}else{

alert(response.getReturnValue().errorMessage);

}

} else if(state == "ERROR"){

alert('Error in calling server side action');

}

});

$A.enqueueAction(action);

},

//Process the selected Farmers record to remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.establishlist");

//temporary array to hold non-selected reocrds

var finalEstablish = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalEstablish.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalEstablish.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalEstablish-' + finalEstablish);

//Now copy the non selected rows to primary array

component.set("v.establishlist", finalEstablish);

},

saveMTCList: function(component, event, helper) {

var estList = component.get("v.establishlist");

//console.log('@@##',estList,'length@@',estList.length);

for(var i=0;i<estList.length;i++){

estList[i].ConcatValue = estList[i].State\_Code\_\_c.concat(estList[i].Location\_Code\_\_c);

}

console.log('estList@@',estList);

var valueArr = estList.map(function(item){ return item.ConcatValue });

var isDuplicate = valueArr.some(function(item, idx){

return valueArr.indexOf(item) != idx

});

if(isDuplicate){

// alert('duplicate value found, go correct');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message":"Hay 2 filas con el mismo valor. Favor de quitar 1 fila y entregar",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "sticky"

});

toastEvent.fire();

}else{

// alert('no duplicate values found, go ahead');

for(var i=0;i<estList.length;i++){

delete estList[i].ConcatValue;

}

var action = component.get("c.saveMTCEstablishForm");

action.setParams({

"Establist": component.get("v.establishlist"),

});

console.log('action###'+action);

//Setting the Callback

action.setCallback(this,function(response){

//get the response state

var state = response.getState();

var res = response.getReturnValue().isSuccessName;

component.set("v.respValue", res);

//component.set("v.showModal", true);

var rlt = component.get("v.respValue");

//document.getElementById("rlt").style.color = "#ff0000";

// alert(rlt);

console.log('@@@@result'+rlt);

//check if result is successfull

if(state == "SUCCESS"){

//Reset Form

console.log(response.getReturnValue());

if(!response.getReturnValue().isError){

if(response.getReturnValue().isDuplicate){

// alert('MTC Record is duplicate with following records'+ response.getReturnValue().isDuplicateName);

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message":"Su MTC ya está creado, continúe con el formulario de Participante : "+ response.getReturnValue().isDuplicateName,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "sticky"

});

toastEvent.fire();

}else{

// alert('Record Created Succesfully'+ response.getReturnValue().isSuccessName);

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "éxito!",

"message": "Registro MTC guardado con éxito y tu código MTC es : "+ response.getReturnValue().isSuccessName ,

duration: '20000',

key: "info\_alt",

type: "success",

mode: "sticky"

});

toastEvent.fire();

//window.location.href='/s/thank-you-page';

//window.location.reload();

$A.get('e.force:refreshView').fire();

}

}else{

// alert(response.getReturnValue().errorMessage);

}

} else if(state == "ERROR"){

alert('Error in calling server side action');

}

});

//adds the server-side action to the queue

$A.enqueueAction(action);

// component.set("v.showModal", true);

}

console.log('after deleting@@@',JSON.stringify(component.get("v.establishlist")));

console.log('after delete estList@@@',estList);

}

})

**Css:**

.THIS .date .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .date .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"Utilice el siguiente formato: DD/MM/AAAA";

}

.THIS .date .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .message .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .message .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"Ingrese un valor válido";

}

.THIS .message .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS.toastMessage.forceActionsText{

font-weight: bolder !important;

}

.THIS .Stateno .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 0px;

}

.THIS .Stateno .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 0;

top: 0;

content:"es un campo numérico";

}

.THIS .stateno .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "";

top: -3.5px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS .button {

background-color: #04AA6D;

border: 1px ;

margin: 25px 50px 75px 100px;

color: white;

padding: 10px;

text-align: left;

text-decoration: none;

display: inline-block;

font-size: 10px;

margin: 4px 2px;

cursor: pointer;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public without sharing class MTC\_EstablishedController {

@AuraEnabled

public static ResponseWrapper saveMTCEstablishForm(list<MTCs\_Established\_\_c> Establist){

set<string> estcode =new set<string>();

list<string> SucString = new list<string>();

//List<MTCs\_Established\_\_c> MTCList = new List<MTCs\_Established\_\_c>();

ResponseWrapper wrap = new ResponseWrapper();

try{

// MTCList.add(Establist);

wrap = checkDuplicateRecord(Establist);

if(wrap.isDuplicate){

return wrap;

}else{

//Database.SaveResult[] lsr = Database.insert(Establist, false);

system.debug('Establist Values' +Establist);

insert Establist;

wrap.isError= false;

for(MTCs\_Established\_\_c est : Establist){

estcode.add(est.ID);

system.debug('for ID:'+ estcode);

}

list<MTCs\_Established\_\_c> MTCSucList = [SELECT Id,Name,State\_Code\_\_c,Location\_Code\_\_c,MTC\_Code\_\_c from MTCs\_Established\_\_c where Id IN:estcode];

for(MTCs\_Established\_\_c MTCSuc: MTCSucList){

// wrap.isDuplicateName = MTC.MTC\_Code\_\_c;

if(MTCSuc.MTC\_Code\_\_c != Null){

SucString.add(MTCSuc.MTC\_Code\_\_c);

system.debug('stringggg'+SucString);

}

}

wrap.isSuccessName = SucString;

return wrap;

}

} catch (Exception ex){

wrap.isError = true;

wrap.errorMessage = ex.getMessage();

return wrap;

}

}

/\*New code added \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@AuraEnabled

public static ResponseWrapper insertData(String strfromle){

set<string> estcode =new set<string>();

list<string> SucString = new list<string>();

String returnresponse ='';

List<MTCs\_Established\_\_c> ListtoInsert = new List<MTCs\_Established\_\_c>();

system.debug('@@1strfromle = ' + strfromle);

list<fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

ResponseWrapper wrap = new ResponseWrapper();

system.debug('@@2datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

// RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag input Form' limit 1];

for(fieldWrapper wrapper: datalist){

MTCs\_Established\_\_c far1 =new MTCs\_Established\_\_c();

system.debug('@@3wrapper'+wrapper);

far1.State\_Code\_\_c = wrapper.CodigodelEstado;

far1.Location\_Code\_\_c =wrapper.CodigodeLocalizacion;

far1.Location\_Name\_\_c =wrapper.Nombredellugar;

far1.Name\_of\_MTC\_\_c = wrapper.NombredeMTC;

far1.Date\_Operational\_\_c =wrapper.Fechadeapertura;

ListtoInsert.add(far1);

}

//Added below by Dhivya. Also changed the return datatype to ResponseWrapper from string

try{

// MTCList.add(Establist);

wrap = checkDuplicateRecord(ListtoInsert);

if(wrap.isDuplicate){

return wrap;

}else{

//Database.SaveResult[] lsr = Database.insert(Establist, false);

system.debug('Establist Values' +ListtoInsert);

insert ListtoInsert;

wrap.isError= false;

for(MTCs\_Established\_\_c est : ListtoInsert){

estcode.add(est.ID);

system.debug('for ID:'+ estcode);

}

list<MTCs\_Established\_\_c> MTCSucList = [SELECT Id,Name,State\_Code\_\_c,Location\_Code\_\_c,MTC\_Code\_\_c from MTCs\_Established\_\_c where Id IN:estcode];

for(MTCs\_Established\_\_c MTCSuc: MTCSucList){

// wrap.isDuplicateName = MTC.MTC\_Code\_\_c;

if(MTCSuc.MTC\_Code\_\_c != Null){

SucString.add(MTCSuc.MTC\_Code\_\_c);

system.debug('stringggg'+SucString);

}

}

wrap.isSuccessName = SucString;

return wrap;

}

} catch (Exception ex){

wrap.isError = true;

wrap.errorMessage = ex.getMessage();

return wrap;

}

/\* if(ListtoInsert.size() > 0){

// try {

// wrap = checkDuplicateRecord(ListtoInsert);

//if(wrap.isDuplicate){

// return wrap;

// system.debug('>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>null>>>>>>>>>>>>>>>>>>>>.');

///}else{

insert ListtoInsert;

returnresponse = 'SUCCESS';

//}

}

//catch(Exception ex){

//returnresponse = 'ERROR';

//}

return returnresponse;commented by Dhivya\*/

}

//return returnresponse;

// }

public class fieldWrapper{

public String CodigodelEstado;

public String CodigodeLocalizacion;

public String Nombredellugar;

public String NombredeMTC;

public String Fechadeapertura;

} /\*

\* Check for Duplicate record in salesforce and return duplicate recordIds.

\* \*/

public static ResponseWrapper checkDuplicateRecord(List<Sobject> recList){

Datacloud.FindDuplicatesResult[] results = Datacloud.FindDuplicates.findDuplicates(recList);

Boolean foundDuplicate = false;

List<id> matchRecordsId = new List<id>();

for (Datacloud.FindDuplicatesResult findDupeResult : results) {

for (Datacloud.DuplicateResult dupeResult : findDupeResult.getDuplicateResults()) {

for (Datacloud.MatchResult matchResult : dupeResult.getMatchResults()) {

for (Datacloud.MatchRecord matchRecord : matchResult.getMatchRecords()) {

foundDuplicate = true;

//System.debug(matchRecord.getRecord().);

matchRecordsId.add(matchRecord.getRecord().Id);

}

}

}

}

list<MTCs\_Established\_\_c> MTCNewList = [SELECT Id,Name,State\_Code\_\_c,Location\_Code\_\_c,MTC\_Code\_\_c from MTCs\_Established\_\_c where Id IN:matchRecordsId];

list<string> DupString = new list<string>();

ResponseWrapper wrap = new ResponseWrapper();

if(foundDuplicate){

wrap.isDuplicate = true;

for(MTCs\_Established\_\_c MTC: MTCNewList){

// wrap.isDuplicateName = MTC.MTC\_Code\_\_c;

if(MTC.MTC\_Code\_\_c != Null){

DupString.add(MTC.MTC\_Code\_\_c);

system.debug('stringggg'+DupString);

}

}

wrap.isDuplicateName = DupString;

wrap.duplicateRecordId = String.valueOf(matchRecordsId);

}else{

wrap.isDuplicate = false;

wrap.duplicateRecordId = '';

}

return wrap;

}

public class ResponseWrapper{

@auraEnabled

public Boolean isDuplicate{get;set;}

@auraEnabled

public list<String> isDuplicateName{get;set;}

@auraEnabled

public list<String> isSuccessName{get;set;}

@auraEnabled

public String duplicateRecordId{get;set;}

@auraEnabled

public Boolean isError{get;set;}

@auraEnabled

public String errorMessage{get;set;}

}

}

**TestClass:**

@isTest

public class MTC\_EstablishedControllerTest {

@istest

static void createData(){

list<MTCs\_Established\_\_c> mtcList = new list<MTCs\_Established\_\_c>();

// Mtc record Creation 1

MTCs\_Established\_\_c mtc1 = new MTCs\_Established\_\_c();

mtc1.State\_Code\_\_c = '84';

mtc1.Location\_Code\_\_c = '29';

mtc1.Location\_Name\_\_c = 'test';

mtc1.Name\_of\_MTC\_\_c = 'test';

mtc1.Date\_Operational\_\_c = '24/02/2022';

insert mtc1;

// Mtc record Creation 2

MTCs\_Established\_\_c mtc2 = new MTCs\_Established\_\_c();

mtc2.State\_Code\_\_c = '84';

mtc2.Location\_Code\_\_c = '29';

mtc2.Location\_Name\_\_c = 'test';

mtc2.Name\_of\_MTC\_\_c = 'test';

mtc2.Date\_Operational\_\_c = '24/02/2022';

mtcList.add(mtc2);

if (Datacloud.FindDuplicates.findDuplicates(mtcList).size() == 0) {

// If the new Mtc's doesn't have duplicates, insert it.

insert(mtcList);

}

// Mtc record Creation 3

MTCs\_Established\_\_c mtc3 = new MTCs\_Established\_\_c();

mtc3.State\_Code\_\_c = '14';

mtc3.Location\_Code\_\_c = '44';

mtc3.Location\_Name\_\_c = 'test';

mtc3.Name\_of\_MTC\_\_c = 'test';

mtc3.Date\_Operational\_\_c = '24/02/2022';

insert mtc3;

list<MTCs\_Established\_\_c> MTCSucList = [SELECT Id,Name,State\_Code\_\_c,Location\_Code\_\_c,MTC\_Code\_\_c from MTCs\_Established\_\_c where MTC\_Code\_\_c='1444'];

System.assertEquals(1, MTCSucList.size());

MTC\_EstablishedController.saveMTCEstablishForm(MTCSucList);

MTC\_EstablishedController.checkDuplicateRecord(mtcList);

}

@istest

static void createCSVData(){

String returnresponse ='';

list<MTCs\_Established\_\_c> mtcList = new list<MTCs\_Established\_\_c>();

// Mtc record Creation 1

MTCs\_Established\_\_c mtc1 = new MTCs\_Established\_\_c();

mtc1.State\_Code\_\_c = '84';

mtc1.Location\_Code\_\_c = '29';

mtc1.Location\_Name\_\_c = 'test';

mtc1.Name\_of\_MTC\_\_c = 'test';

mtc1.Date\_Operational\_\_c = '24/02/2022';

insert mtc1;

// Mtc record Creation 2

MTCs\_Established\_\_c mtc2 = new MTCs\_Established\_\_c();

mtc2.State\_Code\_\_c = '84';

mtc2.Location\_Code\_\_c = '29';

mtc2.Location\_Name\_\_c = 'test';

mtc2.Name\_of\_MTC\_\_c = 'test';

mtc2.Date\_Operational\_\_c = '24/02/2022';

mtcList.add(mtc2);

if (Datacloud.FindDuplicates.findDuplicates(mtcList).size() == 0) {

// If the new Mtc's doesn't have duplicates, insert it.

insert(mtcList);

}

// Mtc record Creation 3

MTCs\_Established\_\_c mtc3 = new MTCs\_Established\_\_c();

mtc3.State\_Code\_\_c = '14';

mtc3.Location\_Code\_\_c = '44';

mtc3.Location\_Name\_\_c = 'test';

mtc3.Name\_of\_MTC\_\_c = 'test';

mtc3.Date\_Operational\_\_c = '24/02/2022';

insert mtc3;

list<MTCs\_Established\_\_c> MTCSucList = [SELECT Id,Name,State\_Code\_\_c,Location\_Code\_\_c,MTC\_Code\_\_c from MTCs\_Established\_\_c where MTC\_Code\_\_c='1444'];

System.assertEquals(1, MTCSucList.size());

MTC\_EstablishedController.insertData(JSON.serialize(MTCSucList));

MTC\_EstablishedController.checkDuplicateRecord(mtcList);

}

}

**Project Participant cmp:**

<aura:component controller="Project\_Participant\_Controller" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Attribute declaration-->

<aura:attribute name="ContactList" type="Contact[]"/>

<aura:attribute name = "checkBoxField" type = "Boolean" default = "false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<aura:attribute name="mtc" type="MTCs\_Established\_\_c" default="{'sObjectType' : 'MTCs\_Established\_\_c'}"/>

<!--Attribute declaration for picklistvalues-->

<aura:attribute name="IndividualTypeMap" type="Map"/>

<aura:attribute name="options\_IndividualType" type="List" default="[]"/>

<aura:attribute name="GenderMap" type="Map"/>

<aura:attribute name="options\_Gender" type="List" default="[]"/>

<aura:attribute name="selectedParticipant" type="List"/>

<!--<aura:attribute name="contactLst" type="list" /> -->

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="selectedRows" type="List"/>

<aura:attribute name="openState" type="String" default="closed" />

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!--Action method/doInit-->

<aura:handler event="aura:waiting" action="{!c.showSpinner}"/>

<aura:handler event="aura:doneWaiting" action="{!c.hideSpinner}"/>

<aura:handler name="init" value="{!this}" action="{!c.doInit}"/>

<aura:handler name="init" value="{!this}" action="{!c.doGender}"/>

<aura:html tag="style">

.slds-col--padded.contentRegion.comm-layout-column{

width:1300px !important;

margin-left:-80px !important;

}

</aura:html>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 70rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario de Registro de Participantes</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

<!--lightning components for designing UI-->

<div class="slds-m-around--xxx--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!--<img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Arial; color:green;">Formulario de Registro de Participantes</span>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead >

<tr class="slds-text-title\_caps" >

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<center> <div class="slds-cell-wrap" >S.No</div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="MTC Code">Código<div> MTC</div>

<div class="requiredclass"><abbr class="slds-required" >\*</abbr></div>

<div class="help"><lightning:helptext content="Elija una opción de la lista en lugar de escribirlo" class="customIcon"/></div>

</div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Participant Number">Número de participante<abbr class="slds-required" >\*</abbr>

<lightning:helptext content="Ingrese un número de 4 dígitos únicamente. No se permite Texto" class="customIcon"/>

</div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="First Name">Apellido<abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col" required="true">

<center><div class="slds-cell-wrap" title="Last Name">Nombre<abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Phone">Genero <abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Age">Edad<abbr class="slds-required" >\*</abbr></div></center>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="State">Estado<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<center><div class="slds-cell-wrap" title="Individual Type" >Tipo de individual<abbr class="slds-required" >\*</abbr>

<lightning:helptext content="¿Cómo se ve en la comunidad?" class="customIcon"/>

</div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Phone">Numero de teléfono<abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="State">Estado<abbr class="slds-required" >\*</abbr></div></center>

</th>

<!--<th scope="col">

<div class="slds-truncate" title="Action">Action</div>

</th>-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.ContactList}" var="con" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkbox" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<c:MTC\_LookUp

objectAPIName="MTCs\_Established\_\_c"

returnFields="['MTC\_Code\_\_c']"

queryFields="['MTC\_Code\_\_c']"

selectedId ="{!con.MTC\_Code\_\_c}"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<td>

<lightning:input name="ParticipantNumber" type="text" maxlength="4" minlength="4" value="{!con.Participant\_Number\_\_c}" pattern="[0-9]\*" class="participantno"/>

</td>

<td>

<lightning:input name="ConFirstName" type="text" maxlength="50" value="{!con.FirstName}" />

</td>

<td>

<lightning:input name="ConLastName" type="text" maxlength="50" value="{!con.LastName}" />

</td>

<td>

<lightning:combobox type="Gender" value="{!con.Genero\_\_c}" options="{!v.options\_Gender}" placeholder="" />

</td>

<td>

<lightning:input name="ConAge" type="Age" maxlength="4" value="{!con.Age\_\_c}" />

</td>

<td>

<lightning:combobox type="Individual Type" value="{!con.TIPO\_DE\_INDIVIDUAL\_\_c}" options="{!v.options\_IndividualType}" placeholder="" />

<!-- <lightning:helptext content="How do you see yourself in the community?" class="customIcon"/> -->

</td>

<td>

<lightning:input name="ConPhone" type="phone" minlength= "10" maxlength="10" value="{!con.Phone}" pattern="[0-9]\*" class="phoneno" />

</td>

<td>

<lightning:input name="ConState" type="State" maxlength="20" value="{!con.State\_Region\_Province\_\_c}" />

</td>

<!--<td>

<a onclick="{!c.removeRow}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td> -->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

</aura:component>

**Controller:**

({

//calling serverside/apex controller for getting picklist values

doInit : function(component, event, helper) {

//alert('9');

helper.addContactRecord(component, event);

var action = component.get("c.getIndividualTypeMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var IndividualTypeMap = [];

var IndividualTypeOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

IndividualTypeMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

IndividualTypeOptions.push(state);

//alert(IndividualTypeOptions);

//system.debug('IndividualTypeOptions'+JSON.Stringfy(IndividualTypeOptions));

}

component.set("v.IndividualTypeMap", IndividualTypeMap);

component.set("v.options\_IndividualType", IndividualTypeOptions);

}

});

$A.enqueueAction(action);

},

//Select all records

handleSelectAllRecords: function(component, event, helper) {

var checked = event.getSource().get('v.checked');

if(component.find('checkbox').length > 1) {

var selectedRows = component.get('v.selectedRows');

component.find('checkbox').forEach(item => {

item.set('v.checked',checked);

selectedRows.push(item.get('v.accesskey'));

});

component.set('v.selectedRows',selectedRows);

}

else {

component.find('checkbox').set('v.checked',checked);

component.set('v.selectedRows',[]);

}

},

//Process the selected records to remove

handleSelectedRecords: function(component, event, helper) {

var selectedContacts = [];

var checkvalue = component.find("selectedContact");

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == true) {

selectedContacts.push(checkvalue.get("v.text"));

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == true) {

selectedContacts.push(checkvalue[i].get("v.text"));

}

}

}

console.log('selectedContacts-' + selectedContacts);

},

doGender : function(component, event, helper) {

//helper.addContactRecord(component, event);

var action = component.get("c.GenderMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var GenderMap = [];

var GenderOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

GenderMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

GenderOptions.push(state);

//alert(IndividualTypeOptions);

//system.debug('IndividualTypeOptions'+JSON.Stringfy(IndividualTypeOptions));

}

component.set("v.GenderMap", GenderMap);

component.set("v.options\_Gender", GenderOptions);

}

});

$A.enqueueAction(action);

},

//add new row method

addRow: function(component, event, helper) {

helper.addContactRecord(component, event);

},

//remove row method(single row)

removeRow1: function(component, event, helper) {

//Get the Farmer list

var ContactList = component.get("v.ContactList");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

ContactList.splice(index, 1);

component.set("v.ContactList", ContactList);

},

getAlert : function(cmp, evt) {

//var selectedEventId = event.target.id;

//alert("Are you sure, you want to delete?");

},

getDuplicateAlert : function(cmp, evt) {

//alert("Dear user , No Duplicates are Allowed !");

},

handleSelectedRows : function(component, event, helper) {

var selectedRows = component.get('v.selectedRows');

var index = event.getSource().get('v.accesskey');

var checked = event.getSource().get('v.checked');

if(checked) {

if($A.util.isEmpty(selectedRows)) selectedRows.push(index);

else

if(!selectedRows.includes(index)) selectedRows.push(index)

}

else {

selectedRows.forEach(function(rowValue,rowIndex) {

//alert(rowValue+' '+index)

if(rowValue == index) selectedRows.splice(rowIndex,1)

});

}

component.set('v.selectedRows',selectedRows);

},

//select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.ContactList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkbox = component.find("checkbox");

if(checkvalue == true){

for(var i=0; i<checkbox.length; i++){

checkbox[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkbox.length; i++){

checkbox[i].set("v.value",false);

}

}

},

showSpinner: function(component, event, helper) {

component.set("v.Spinner", true);

},

hideSpinner : function(component,event,helper){

component.set("v.Spinner", false);

},

save: function(component, event, helper) {

if (helper.validateContactList(component, event)) {

helper.saveContactList(component, event);

}

},

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

// Showing the values in Popup while uploading files in both .csv and .xlsx format

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("CodigoMTC", "Numerodeparticipante", "Apellido", "Nombre" , "Genero","Edad","TipoDeIndividual","Numerodetelefono", "Estado");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

// alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

//alert('csv text' + text);

console.log(text);

textdata = text;

var table = document.createElement("table");

//alert(' csv table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

// alert('csv rows>>'+ rows);

var arr=[];

rows.pop();

//alert(' csv rows>>' + rows);

var headers = rows[0].split(',');

//alert('csv headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('csv data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

// alert('csv arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('xlsx onload xlsreader');

const headers2 = new Array("CodigoMTC", "Numerodeparticipante", "Apellido", "Nombre" , "Genero","Edad","TipoDeIndividual","Numerodetelefono", "Estado");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

// alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

//alert('xlsx table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

// Validating and Saving the records while uploading both .csv and .xlsx format

readCSV: function(component, event, helper) {

const headers2 = new Array("CodigoMTC", "Numerodeparticipante", "Apellido", "Nombre" , "Genero","Edad","TipoDeIndividual","Numerodetelefono", "Estado");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert('read csv'+ reader);

var text = reader.result; /\*Get the data stored in file\*/

// alert('read csv'+ text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

// alert('readCsv rows>>'+ rows);

var arr=[];

rows.pop();

// alert('readCsv rows>>' + rows);

var headers = rows[0].split(',');

//alert('readcsv headers>>' + headers);

var headers = rows[0].split(',');

// alert('readCsv headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('readCsv data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('readCsv arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].CodigoMTC=="" ||

data1[k].Numerodeparticipante==""||

data1[k].Apellido=="" ||

data1[k].Nombre=="" ||

data1[k].Genero=="" ||

data1[k].Edad=="" ||

data1[k].TipoDeIndividual=="" ||

data1[k].Numerodetelefono=="" ||

data1[k].Estado==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

// alert('jstonstr>>' + jsonstr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var CodigoMTC;

var Numerodeparticipante;

var Apellido;

var Nombre;

var Genero;

var Edad;

var TipoDeIndividual;

var Numerodetelefono;

var Estado;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

// alert('readCsv xlsx rows'+ rows);

arr.pop();

// alert('readCsv xlsx arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

// alert('readCsv xlsx headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('readCsv xlsx Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].CodigoMTC=="" ||

data1[k].Numerodeparticipante==""||

data1[k].Apellido=="" ||

data1[k].Nombre=="" ||

data1[k].Genero=="" ||

data1[k].Edad=="" ||

data1[k].TipoDeIndividual=="" ||

data1[k].Numerodetelefono=="" ||

data1[k].Estado==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var CodigoMTC;

var Numerodeparticipante;

var Apellido;

var Nombre;

var Genero;

var Edad;

var TipoDeIndividual;

var Numerodetelefono;

var Estado;

var infolst = [];

var jsonstr = JSON.stringify(rows);

// alert(' xlsx jsonstr>>'+ jsonstr )

console.log('jsonstr' +jsonstr);

//alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

CSV2JSON: function (component,csv) {

// console.log('Incoming csv = ' + csv);

//var array = [];

var arr = [];

arr = csv.split('\n');

//console.log('Array = '+array);

// console.log('arr = '+arr);

arr.pop();

var jsonObj = [];

var headers = arr[0].split(',');

for(var i = 1; i < arr.length; i++) {

var data = arr[i].split(',');

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

jsonObj.push(obj);

}

/\* var json = JSON.stringify(jsonObj);

console.log('json = '+ json);

return json;\*/

var isValid = true;

var data1 = jsonObj;

for( var k=0;k<jsonObj.length;k++){

console.log(data1);

if(data1[k].CodigoMTC=="" ||

data1[k].Numerodeparticipante==""||

data1[k].Apellido=="" ||

data1[k].Nombre=="" ||

data1[k].Genero=="" ||

data1[k].Edad=="" ||

data1[k].TipoDeIndividual=="" ||

data1[k].Numerodetelefono=="" ||

data1[k].Estado==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

/\* isValid = false;

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

// return isValid;

}

}

return isValid;\*/

var json = JSON.stringify(data1);

console.log('json = '+ json);

return json;

},

CreateContact : function (component,jsonstr){

console.log('jsonstr' + jsonstr);

var action = component.get('c.insertData');

//alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this,function(response){

//get the response state

var state = response.getState();

//alert(state);

var res = response.getReturnValue().isSuccessName;

component.set("v.respValue", res);

//component.set("v.showModal", true);

var rlt = component.get("v.respValue");

//document.getElementById("rlt").style.color = "#ff0000";

// alert(rlt);

console.log('@@@@result'+rlt);

//check if result is successfull

if(state == "SUCCESS"){

//Reset Form

console.log(response.getReturnValue());

//alert(response.getReturnValue());

if(!response.getReturnValue().isError){

if(response.getReturnValue().isDuplicate){

// alert('Contact Record is duplicate with following records'+ response.getReturnValue().isDuplicateName);

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Es posible que esté creando un registro duplicado. Puede eliminar la fila" + response.getReturnValue().isDuplicateName ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}else{

// alert('Record Created Succesfully'+ response.getReturnValue().isSuccessName);

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "éxito!",

"message": "Los registros de contacto fueron guardados : "+ response.getReturnValue().isSuccessName ,

duration: '3000',

key: "info\_alt",

type: "success",

mode: "sticky"

});

toastEvent.fire();

//alert('Record Created Succesfully');

$A.get('e.force:refreshView').fire();

}

}else{

alert(response.getReturnValue().errorMessage);

}

} else if(state == "ERROR"){

alert('Error in calling server side action');

}

});

//adds the server-side action to the queue

$A.enqueueAction(action);

},

addContactRecord: function(component, event) {

//get the Contact List from component

var ContactList = component.get("v.ContactList");

//Add New Contact Record

ContactList.push({

'sobjectType': 'Contact',

'FirstName': '',

'LastName': '',

'Genero\_\_c':'',

'Age\_\_c' : '',

'State\_Region\_Province\_\_c':'',

'Phone': '',

'MTC\_Code\_\_c':'',

'ConcatValue':'',

'Participant\_Number\_\_c':''

});

component.set("v.ContactList", ContactList);

console.log('ContactList@@',ContactList);

},

//checking validation for all required fields

validateContactList: function(component, event) {

/\*/Validate all Contact records

var action = component.get("c.duplicateContacts");

action.setParams({

"ConsList": component.get("v.ContactList"),

});\*/

var isValid = true;

var ContactList = component.get("v.ContactList");

for (var i = 0; i < ContactList.length; i++) {

//var cons = [Select LastName,Phone from contact];

if (ContactList[i].FirstName == '' ||

ContactList[i].LastName == '' ||

ContactList[i].Genero\_\_c == '' ||

ContactList[i].Age\_\_c == '' ||

ContactList[i].State\_Region\_Province\_\_c == '' ||

ContactList[i].Phone.length != 10 ||

ContactList[i].Phone == '' ||

ContactList[i].MTC\_Code\_\_c == '' ||

ContactList[i].Participant\_Number\_\_c == '') {

isValid = false;

// alert('Las celdas obligatorias no pueden estar en blanco en la fila' + (i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila" + (i+1) ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "sticky"

});

toastEvent.fire();

}

}

return isValid;

},

//process the selected records to remove

removeRow: function(component, event) {

var selectedContacts = component.get("v.ContactList");

//temporary array to hold non-selected reocrds

var finalContacts = [];

var checkvalue = component.find("checkbox");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalContacts.push(selectedContacts[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalContacts.push(selectedContacts[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalContacts-' + finalContacts);

//Now copy the non selected rows to primary array

component.set("v.ContactList", finalContacts);

},

//final method to create record in server side

saveContactList: function(component, event, helper) {

var estList = component.get("v.ContactList");

//console.log('@@##',estList,'length@@',estList.length);

for(var i=0;i<estList.length;i++){

estList[i].ConcatValue = estList[i].MTC\_Code\_\_c.concat(estList[i].Participant\_Number\_\_c);

}

console.log('estList@@',estList);

var valueArr = estList.map(function(item){ return item.ConcatValue });

var isDuplicate = valueArr.some(function(item, idx){

return valueArr.indexOf(item) != idx

});

if(isDuplicate){

// alert('duplicate value found, go correct');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message":"Hay 2 filas con el mismo valor. Favor de quitar 1 fila y entregar",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "sticky"

});

toastEvent.fire();

}else{

//Call Apex class and pass Contact list parameters

var action = component.get("c.saveContacts");

action.setParams({

"ConList": component.get("v.ContactList"),

});

console.log('action###'+action);

//Setting the Callback

action.setCallback(this,function(response){

//get the response state

var state = response.getState();

//check if result is successfull

if(state == "SUCCESS"){

//Reset Form

console.log(response.getReturnValue());

if(!response.getReturnValue().isError){

if(response.getReturnValue().isDuplicate){

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Es posible que esté creando un registro duplicado. Puede omitir este formulario." + response.getReturnValue().isDuplicateName ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}else{

// alert('Record Created Succesfully'+ response.getReturnValue().isSuccessName);

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "éxito!",

"message": "Los registros de contacto fueron guardados : "+ response.getReturnValue().isSuccessName ,

duration: '3000',

key: "info\_alt",

type: "success",

mode: "sticky"

});

toastEvent.fire();

//alert('Record Created Succesfully');

$A.get('e.force:refreshView').fire();

}

}else{

// alert(response.getReturnValue().errorMessage);

}

} else if(state == "ERROR"){

alert('Error in calling server side action');

}

});

//adds the server-side action to the queue

$A.enqueueAction(action);

}

},

})

**Css:**

.THIS .slds-truncate {

text-align: center;

}

.THIS .slds-icon-utility-delete.slds-icon\_container{

padding-top: 14px;

padding-left: 18px;

}

.THIS .phoneno .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: -12px;

}

.THIS .phoneno .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 0;

top: 0;

content:"Numero de telefono requiere diez numero";

}

.THIS .phoneno .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "";

top: -3.5px;

}

.THIS .slds-combobox\_\_form-element {

width: 100px;

}

.THIS .participantno .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: -12px;

}

.THIS .participantno .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 0;

top: 0;

content:"es un campo numérico";

}

.THIS .participantno .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "";

top: -3.5px;

}

.THIS .help{

margin-right: -76px;

margin-top: -20px;

}

.THIS .requiredclass{

margin-right: -40px;

margin-top: -12px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS .button {

background-color: #04AA6D;

border: 1px ;

margin: 25px 50px 75px 100px;

color: white;

padding: 10px;

text-align: left;

text-decoration: none;

display: inline-block;

font-size: 10px;

margin: 4px 2px;

cursor: pointer;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public without sharing class Project\_Participant\_Controller {

@AuraEnabled

public static ResponseWrapper saveContacts(List<Contact> ConList){

set<string> estcode =new set<string>();

list<string> SucString = new list<string>();

ResponseWrapper wrap = new ResponseWrapper();

try{

wrap = checkDuplicateRecord(ConList);

if(wrap.isDuplicate){

return wrap;

}else{

insert ConList;

wrap.isError= false;

for(Contact est : ConList){

estcode.add(est.ID);

system.debug('for ID:'+ estcode);

}

list<Contact> ConSucList = [SELECT Id,Name,Project\_Participant\_Code\_\_c from Contact where Id IN:estcode];

for(Contact ConSuc: ConSucList){

// wrap.isDuplicateName = ConSuc.Project\_Participant\_Code\_\_c;

if(ConSuc.Project\_Participant\_Code\_\_c != Null){

SucString.add(ConSuc.Project\_Participant\_Code\_\_c);

system.debug('stringggg'+SucString);

}

}

wrap.isSuccessName = SucString;

return wrap;

}

} catch (Exception ex){

wrap.isError = true;

wrap.errorMessage = ex.getMessage();

return wrap;

}

}

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static ResponseWrapper insertData(String strfromle){

set<string> estcode =new set<string>();

list<string> SucString = new list<string>();

String returnresponse ='';

List<Contact> ListtoInsert = new List<Contact>();

system.debug('strfromle = ' + strfromle);

list< fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

ResponseWrapper wrap = new ResponseWrapper();

//system.debug('datalistSize = ' + datalist.size());

// RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Extension Agent Participant Training Form' limit 1];

for(fieldWrapper wrapper: datalist){

Contact Cont =new Contact();

system.debug('wrapper'+wrapper);

//far1.RecordTypeId =RT.Id;

cont.Code1\_\_c = Wrapper.CodigoMTC;

cont.Participant\_Number\_\_c = wrapper.Numerodeparticipante;

cont.FirstName = wrapper.Apellido;

cont.LastName = Wrapper.Nombre;

cont.Genero\_\_c = Wrapper.Genero;

cont.Age\_\_c = wrapper.Edad;

cont.TIPO\_DE\_INDIVIDUAL\_\_c = wrapper.TipoDeIndividual;

cont.Phone= wrapper.Numerodetelefono;

Cont.State\_Region\_Province\_\_c = wrapper.Estado;

ListtoInsert.add(cont);

}

//Added below by Dhivya. Also changed the return datatype to ResponseWrapper from string

try{

// MTCList.add(Establist);

wrap = checkDuplicateRecord(ListtoInsert);

if(wrap.isDuplicate){

return wrap;

}else{

//Database.SaveResult[] lsr = Database.insert(Establist, false);

system.debug('Establist Values' +ListtoInsert);

insert ListtoInsert;

wrap.isError= false;

for(Contact est : ListtoInsert){

estcode.add(est.ID);

system.debug('for ID:'+ estcode);

}

list<Contact> ConSucList = [SELECT Id,Name,MTC\_Code\_\_c,Participant\_Number\_\_c,Project\_Participant\_Code\_\_c from Contact where Id IN:estcode];

for(Contact ConSuc: ConSucList){

// wrap.isDuplicateName = MTC.MTC\_Code\_\_c;

if(ConSuc.Project\_Participant\_Code\_\_c != Null){

SucString.add(ConSuc.Project\_Participant\_Code\_\_c);

system.debug('stringggg'+SucString);

}

}

wrap.isSuccessName = SucString;

return wrap;

}

} catch (Exception ex){

wrap.isError = true;

wrap.errorMessage = ex.getMessage();

return wrap;

}

}

public class fieldWrapper{

public String CodigoMTC;

public String Numerodeparticipante;

public String Apellido;

public String Nombre;

public String Genero;

public Decimal Edad;

public String TipoDeIndividual;

public String Numerodetelefono;

public String Estado;

// public String Code;

}

/\*

\* Check for Duplicate record in salesforce and return duplicate recordIds.

\* \*/

public static ResponseWrapper checkDuplicateRecord(List<Sobject> recList){

Datacloud.FindDuplicatesResult[] results = Datacloud.FindDuplicates.findDuplicates(recList);

Boolean foundDuplicate = false;

List<id> matchRecordsId = new List<id>();

for (Datacloud.FindDuplicatesResult findDupeResult : results) {

//System.debug('debug1'+findDupeResult);

for (Datacloud.DuplicateResult dupeResult : findDupeResult.getDuplicateResults()) {

//System.debug('debug2'+dupeResult);

for (Datacloud.MatchResult matchResult : dupeResult.getMatchResults()) {

//System.debug('debug3'+matchResult);

for (Datacloud.MatchRecord matchRecord : matchResult.getMatchRecords()) {

//System.debug('debug4'+matchRecord);

foundDuplicate = true;

//System.debug(matchRecord.getRecord().);

matchRecordsId.add(matchRecord.getRecord().Id);

}

}

}

}

list<Contact> CondupList = [SELECT Id,Name,Project\_Participant\_Code\_\_c from Contact where Id IN:matchRecordsId];

list<string> DupString = new list<string>();

ResponseWrapper wrap = new ResponseWrapper();

system.debug('foundDuplicate'+foundDuplicate);

if(foundDuplicate){

wrap.isDuplicate = true;

for(Contact Con: CondupList){

// wrap.isDuplicateName = Con.Project\_Participant\_Code\_\_c;

if(Con.Project\_Participant\_Code\_\_c != Null){

DupString.add(Con.Project\_Participant\_Code\_\_c);

DupString.add(Con.Name);

// DupString.add(Con.Project\_Participant\_Code\_\_c);

system.debug('stringggg'+DupString);

}

}

wrap.isDuplicateName = DupString;

wrap.duplicateRecordId = String.valueOf(matchRecordsId);

}else{

wrap.isDuplicate = false;

wrap.duplicateRecordId = '';

}

return wrap;

}

@AuraEnabled

public static Map<String, String> getIndividualTypeMethod(){

Map<String, String> optionsIndividualType = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Contact.TIPO\_DE\_INDIVIDUAL\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsIndividualType.put(ple.getValue(), ple.getLabel());

}

return optionsIndividualType;

}

@AuraEnabled

public static Map<String, String> GenderMethod(){

Map<String, String> optionsGender = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Contact.Genero\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsGender.put(ple.getValue(), ple.getLabel());

}

return optionsGender;

}

public class ResponseWrapper{

@auraEnabled

public Boolean isDuplicate{get;set;}

@auraEnabled

public list<String> isDuplicateName{get;set;}

@auraEnabled

public list<String> isSuccessName{get;set;}

@auraEnabled

public String duplicateRecordId{get;set;}

@auraEnabled

public Boolean isError{get;set;}

@auraEnabled

public String errorMessage{get;set;}

}

}

**TestClass:**

@isTest

public class Project\_Participant\_ControllerTest {

@istest

static void ProjectPartiTest(){

list<String> isDuplicateName;

// Mtc record Creation

MTCs\_Established\_\_c mtc = new MTCs\_Established\_\_c();

mtc.State\_Code\_\_c = '84';

mtc.Location\_Code\_\_c = '29';

mtc.Location\_Name\_\_c = 'test';

mtc.Name\_of\_MTC\_\_c = 'test';

mtc.Date\_Operational\_\_c = '24/02/2022';

insert mtc;

//Contact Creation

list<Contact> ContactList = new list<Contact> ();

contact con1 = new contact();

con1.FirstName='test';

con1.lastname='test123';

con1.Participant\_Number\_\_c='1234';

con1.Genero\_\_c='Mujer';

con1.Age\_\_c=23;

con1.TIPO\_DE\_INDIVIDUAL\_\_c='productor';

con1.Phone='9012746545';

con1.State\_Region\_Province\_\_c='Alaska';

con1.email='test123@gmail.com';

con1.MTC\_Code\_\_c = mtc.Id;

insert con1;

contact con2 = new contact();

con2.FirstName='test';

con2.lastname='test123';

con2.Participant\_Number\_\_c='1234';

con2.Genero\_\_c='Mujer';

con2.Age\_\_c=23;

con2.TIPO\_DE\_INDIVIDUAL\_\_c='productor';

con2.Phone='9012746545';

con2.State\_Region\_Province\_\_c='Alaska';

con2.email='test123@gmail.com';

con2.MTC\_Code\_\_c = mtc.Id;

ContactList.add(con2);

//insert con2;

if (Datacloud.FindDuplicates.findDuplicates(ContactList).size() == 0) {

// If the new contact doesn't have duplicates, insert it.

insert(ContactList);

}

contact con3 = new contact();

con3.FirstName='test4';

con3.lastname='test183';

con3.Participant\_Number\_\_c='8765';

con3.Genero\_\_c='Mujer';

con3.Age\_\_c=23;

con3.TIPO\_DE\_INDIVIDUAL\_\_c='productor';

con3.Phone='9012746545';

con3.State\_Region\_Province\_\_c='Alaska';

con3.email='test126783@gmail.com';

con3.MTC\_Code\_\_c = mtc.Id;

insert con3;

List<Contact> NoDupeContacts = [SELECT Id,Name,Participant\_Number\_\_c,Project\_Participant\_Code\_\_c FROM Contact

WHERE Project\_Participant\_Code\_\_c = '8429 - 8765'];

system.debug('NoDupeContacts@@'+NoDupeContacts);

System.assertEquals(1, NoDupeContacts.size());

Project\_Participant\_Controller.saveContacts(NoDupeContacts);

Project\_Participant\_Controller.checkDuplicateRecord(ContactList);

Map<String, String> getIndividualTypeMethod = Project\_Participant\_Controller.getIndividualTypeMethod();

system.assertNotEquals(Null, getIndividualTypeMethod);

Map<String, String> GenderMethod = Project\_Participant\_Controller.GenderMethod();

system.assertNotEquals(Null, GenderMethod);

}

@istest

static void ProjectPartiCSVTest(){

String returnresponse ='';

list<String> isDuplicateName;

// Mtc record Creation

MTCs\_Established\_\_c mtc = new MTCs\_Established\_\_c();

mtc.State\_Code\_\_c = '84';

mtc.Location\_Code\_\_c = '29';

mtc.Location\_Name\_\_c = 'test';

mtc.Name\_of\_MTC\_\_c = 'test';

mtc.Date\_Operational\_\_c = '24/02/2022';

insert mtc;

//Contact Creation

list<Contact> ContactList = new list<Contact> ();

contact con1 = new contact();

con1.FirstName='test';

con1.lastname='test123';

con1.Participant\_Number\_\_c='1234';

con1.Genero\_\_c='Mujer';

con1.Age\_\_c=23;

con1.TIPO\_DE\_INDIVIDUAL\_\_c='productor';

con1.Phone='9012746545';

con1.State\_Region\_Province\_\_c='Alaska';

con1.email='test123@gmail.com';

con1.MTC\_Code\_\_c = mtc.Id;

insert con1;

contact con2 = new contact();

con2.FirstName='test';

con2.lastname='test123';

con2.Participant\_Number\_\_c='1234';

con2.Genero\_\_c='Mujer';

con2.Age\_\_c=23;

con2.TIPO\_DE\_INDIVIDUAL\_\_c='productor';

con2.Phone='9012746545';

con2.State\_Region\_Province\_\_c='Alaska';

con2.email='test123@gmail.com';

con2.MTC\_Code\_\_c = mtc.Id;

ContactList.add(con2);

//insert con2;

if (Datacloud.FindDuplicates.findDuplicates(ContactList).size() == 0) {

// If the new contact doesn't have duplicates, insert it.

insert(ContactList);

}

contact con3 = new contact();

con3.FirstName='test4';

con3.lastname='test183';

con3.Participant\_Number\_\_c='8765';

con3.Genero\_\_c='Mujer';

con3.Age\_\_c=23;

con3.TIPO\_DE\_INDIVIDUAL\_\_c='productor';

con3.Phone='9012746545';

con3.State\_Region\_Province\_\_c='Alaska';

con3.email='test126783@gmail.com';

con3.MTC\_Code\_\_c = mtc.Id;

insert con3;

List<Contact> NoDupeContacts = [SELECT Id,Name,Participant\_Number\_\_c,Project\_Participant\_Code\_\_c FROM Contact

WHERE Project\_Participant\_Code\_\_c = '8429 - 8765'];

system.debug('NoDupeContacts@@'+NoDupeContacts);

System.assertEquals(1, NoDupeContacts.size());

Project\_Participant\_Controller.insertData(JSON.serialize(NoDupeContacts));

Project\_Participant\_Controller.checkDuplicateRecord(ContactList);

Map<String, String> getIndividualTypeMethod = Project\_Participant\_Controller.getIndividualTypeMethod();

system.assertNotEquals(Null, getIndividualTypeMethod);

Map<String, String> GenderMethod = Project\_Participant\_Controller.GenderMethod();

system.assertNotEquals(Null, GenderMethod);

}

}

**Ag Input cmp:**

<aura:component controller="AG\_Input\_Form1Controller" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Attribute declaration-->

<aura:attribute name="Farmer" type="Farmers\_\_c[]"/>

<aura:attribute name="farm" type="Farmers\_\_c" default="{'sObjectType' : 'Farmers\_\_c'}"/>

<aura:attribute name="Contacts" type="Contact[]"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!--<aura:attribute name="trainerState" type="boolean" default="false"/>-->

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!-- Attribute declaration for picklistValues -->

<aura:attribute name="TypeMap" type="Map"/>

<aura:attribute name="options\_Type" type="List" default="[]"/>

<aura:attribute name="AgeMap" type="Map"/>

<aura:attribute name="options\_Age" type="List" default="[]"/>

<aura:attribute name="GenderMap" type="Map"/>

<aura:attribute name="options\_Gender" type="List" default="[]"/>

<!-- Attribute declaration for LoolupFields-->

<aura:attribute name="FarmerState" type="String"/>

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<!-- <aura:handler name="change" value="{!v.selectedContactId}" action="{!c.handleContactChange}" /> -->

<aura:attribute name="selectedContactId" type="String" />

<!--Action Method/doInit-->

<aura:handler name="init" value="{!this}" action="{!c.doIt}"/>

<!--<aura:handler name="init" value="{!this}" action="{!c.doAge}"/>

<aura:handler name="init" value="{!this}" action="{!c.doGender}"/>-->

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<!-- lightning Components for designing UI-->

<div class="slds-m-around--x--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!-- <img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Times New Roman; color:green;">Formulario De Insumos Agrícolas</span>

</div>

<lightning:layout >

<!-- <lightning:layoutitem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small ">

<lightning:input name="State" label="State" type="text" />

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Estado<abbr class="slds-required" >\*</abbr></label>

<lightning:input label="State" value="{!v.State1\_\_c}"

aura:id="state" required="true" onchange="{!c.captureState}" variant="label-hidden" class="message"/>

</div>

</lightning:layoutitem>-->

<lightning:layoutitem size="6" >

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small ">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Numero de identificación de extensionista<abbr class="slds-required" >\*</abbr></label>

<!-- <lightning:input label="Extension Agent Name" value="{!v.ExtensionAgentId\_\_c}"

aura:id="field" required="true" variant="label-hidden"/> -->

<c:ContactTrainersLookup

objectAPIName="Contact"

label="Contact"

returnFields="['ID\_\_c']"

queryFields="['ID\_\_c']"

selectedId="{!v.selectedContactId}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z" />

</div>

</lightning:layoutitem>

</lightning:layout>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-truncate" >S.No</div>

</th>

<th scope="col">

<div class="slds-truncate" title="Type" align="center" >Numero de identificación <abbr class="slds-required" >\*</abbr></div>

<!--<div class="slds-truncate" title="Person Id" align="center" onkeyup="{!c.onChangeSearchText}" >Person ID<abbr class="slds-required" >\*</abbr>-->

<!--<label class="slds-form-element\_\_label" ></label>-->

<!--<lightning:input aura:id="personID" name="personID" label="" type="search" placeholder="Search Person Id" isLoading="{! v.issearching }" />

<lightning:helptext content="First letter of the InitialName and LastName with Last 5 digit of PhoneNumber (ForEx:VB85432)" class="customIcon"/>

</div>-->

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="Age" align="center" >Edad<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-truncate" title="Gender" align="center" >Genero<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<div class="slds-truncate" title="Type" align="center" >Tipo de insumo<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-truncate" title="Sub-type" align="center">Subtipo de insumo<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-truncate" title="Quantity" align="center">Cantidad<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="Action" align="center">Action</div>

</th>-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.Farmer}" var="far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!--<div class="slds-truncate" align="center" onkeyup="{!c.onChangeSearchText}" >

<label class="slds-form-element\_\_label" ></label>

<lightning:input aura:id="personID" name="personID" onkeyup="{!c.onChangeSearchText}" label="" type="search" placeholder="Search Person Id" isLoading="{! v.issearching }" />-->

<!-- Calling another component for creating lookup -->

<c:ContactFarmerLookUp

objectAPIName="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId="{!far.Contact\_Name\_\_c}"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

<!-- <c:LookupPersonId objectAPIName="contact" IconName="standard:contact" />-->

<!--<lightning:input name="Personid" type="text" value="{!v.farmerId}" />

lightning:input aura:id="personID" name="personID" label="" type="search" placeholder="Search Person Id" isLoading="{! v.issearching }" /-->

</td>

<!-- <td>

<lightning:combobox type="Age" value="{!far.Edad\_\_c}" options="{!v.options\_Age}" placeholder="" />

</td>

<td>

<lightning:combobox type="Gender" value="{!far.Genero\_\_c}" options="{!v.options\_Gender}" placeholder="" />

</td>-->

<td>

<lightning:combobox type="Type" value="{!far.TIPO\_DE\_INSUMO\_\_c}" options="{!v.options\_Type}" placeholder="" />

</td>

<td>

<lightning:input name="SubType" type="text" value="{!far.Sub\_Type\_\_c}" />

</td>

<td>

<lightning:input name="Quantity" type="text" value="{!far.Quantity\_\_c}" />

</td>

<!-- <td>

<a onclick="{!c.removeRow}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td> -->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 70rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario De Insumos Agrícolas</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

</aura:component>

**Controller:**

({

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("ExtensionistaId", "TipoDeInsumo", "SubTipo", "Cantidad" , "Identificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

//alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

// alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

// alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers.length);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

// alert('ExtensionistaId,TipoDeInsumo,SubTipo,Cantidad,Identificacion Se permiten ');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("ExtensionistaId", "TipoDeInsumo", "SubTipo", "Cantidad", "Identificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

// alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

// alert('ExtensionistaId,TipoDeInsumo,SubTipo,Cantidad,Identificacion Se permiten ');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

//calling serverside/apex controller for getting PicklistValues

doIt : function(component, event, helper) {

helper.addFarmerRecord(component, event);

var action = component.get("c.getTypeMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var FarmerTypeMap = [];

var FarmerTypeOptions = [];

for(var key in result){

FarmerTypeMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

FarmerTypeOptions.push(state);

}

component.set("v.TypeMap", FarmerTypeMap);

component.set("v.options\_Type", FarmerTypeOptions);

}

});

$A.enqueueAction(action);

},

//Select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.Farmer");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//adding new row method

addRow: function(component, event,helper) {

helper.addFarmerRecord(component, event);

},

//remove row method (single row)

removeRecord: function(component, event, helper) {

//Get the list

var FarmerList = component.get("v.Farmer");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

FarmerList.splice(index, 1);

component.set("v.Farmer", FarmerList);

},

//final method to create records in server side

save: function(component, event, helper) {

var FarmerList = component.get("v.Farmer");

// checking validation for required fields FarmerState,selectedContactId

//var StateValue =component.get("v.FarmerState");

var ExtAgenrId =component.get("v.selectedContactId");

//if(typeof StateValue == 'undefined' || typeof ExtAgenrId == 'undefined'){

if( typeof ExtAgenrId == 'undefined'){

// alert('Las celdas obligatorias no pueden estar en blanco');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

else{

if (helper.validateFarmerRecords(component, event)) {

var StateValue =component.get("v.FarmerState");

var ExtAgenrId =component.get("v.selectedContactId");

var FarmerList = component.get("v.Farmer");

for (var i = 0; i < FarmerList.length; i++) {

//FarmerList[i].State1\_\_c = StateValue;

FarmerList[i].ExtensionAgentId\_\_c = ExtAgenrId;

}

helper.saveFarmerList(component, event);

}

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

//getting value from the client side

captureState :function(component,event){

var NOTRValue = event.getParam("value");

//alert(NOTGValue);

component.set("v.FarmerState", NOTRValue);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

readCSV: function(component, event, helper) {

const headers2 = new Array("ExtensionistaId", "TipoDeInsumo", "SubTipo", "Cantidad", "Identificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

// alert('ExtensionistaId,TipoDeInsumo,SubTipo,Cantidad,Identificacion Se permiten ');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].ExtensionistaId=="" ||

data1[k].TipoDeInsumo==""||

data1[k].SubTipo=="" ||

data1[k].Cantidad=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var ExtensionistaId ;

var TipoDeInsumo;

var SubTipo;

var Cantidad;

var Identificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

// alert(rows);

arr.pop();

// alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

// alert('ExtensionistaId,TipoDeInsumo,SubTipo,Cantidad,Identificacion Se permiten ');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].ExtensionistaId=="" ||

data1[k].TipoDeInsumo==""||

data1[k].SubTipo=="" ||

data1[k].Cantidad=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var ExtensionistaId ;

var TipoDeInsumo;

var SubTipo;

var Cantidad;

var Identificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

CSV2JSON: function (component,csv) {

// console.log('Incoming csv = ' + csv);

//var array = [];

var arr = [];

arr = csv.split('\n');

//console.log('Array = '+array);

// console.log('arr = '+arr);

arr.pop();

var jsonObj = [];

var headers = arr[0].split(',');

console.log('Headers>>'+ headers);

const headers2 = new Array("ExtensionistaId", "TipoDeInsumo", "SubTipo", "Cantidad", "Identificacion");

console.log('headers2>>'+ headers2 );

/\* var map = new Map();

map.set("headers2", headers2);

alert(map.get("headers2"));

// console.log(map1);\*/

/\* if(headers != map.get("headers2")){

alert('Value mismatched');

return null;

}\*/

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

alert("Values Mismatched");

return null;

//console.log('Values Mismatched');

}

}

for(var i = 1; i < arr.length; i++) {

var data = arr[i].split(',');

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

jsonObj.push(obj);

}

var data1 = jsonObj;

for( var k=0;k<jsonObj.length;k++){

//data1 = JSON.stringify(jsonObj[k]);

//data1 = jsonObj[k];

// data2 = JSON.stringify(data1)

console.log(data1);

if(

data1[k].TipoDeInsumo=="" ||

data1[k].SubTipo==""||

data1[k].Cantidad=="" ||

//data1[k].ExtenAgent==""||

data1[k].Identificacion==""){

//alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

else if(data1[k].ExtensionistaId==""){

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor complete los valores de identificación de extensionista ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var json = JSON.stringify(data1);

console.log('json = '+ json);

return json;

},

CreateContact : function (component,jsonstr){

// console.log('jsonstr' + jsonstr);

// alert('jsonstr'+ jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

// alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

// alert('result callback>>'+ result);

// alert("Records Inserted Succesfully");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

//alert("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

addFarmerRecord : function(component, event){

//get the Farmer list from Component

var FarmerList = component.get("v.Farmer");

//Add New Farmer Record

FarmerList.push({

'sobjectType':'Farmers\_\_c',

'ID\_\_c':'',

//'State1\_\_c':'',

//'Edad\_\_c':'',

//'Genero\_\_c':'',

'TIPO\_DE\_INSUMO\_\_c':'',

'Sub\_Type\_\_c':'',

'Quantity\_\_c':'',

'Contact\_Name\_\_c' : '',

'ExtensionAgentId\_\_c' : ''

});

component.set("v.Farmer",FarmerList);

},

//Process the selected Farmers records to remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.Farmer");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.Farmer", finalFarmers);

},

/\* validateFarmerList:function(component,event){

var action=component.get("c.getPersonID");

action.setParams({

"personId":personId

});

action.setCallback(this,function(response){

var state=response.getState();

if(state === "SUCCESS"){

// alert("state");

component.set("v.PersonId",response.getReturnValue)

//alert("state");

}

else{

alert( "This id do not exist please fill the PROJECT PARTICIPANT INFORMATION FORM.");

}

});

$A.enqueueAction(action);

},\*/

//checking Validation for all required fields

validateFarmerRecords: function(component, event) {

//Validate all Farmer records

var isValid = true;

var farmersList = component.get("v.Farmer");

console.log('farmersList'+farmersList);

for (var i = 0; i < farmersList.length; i++) {

if (farmersList[i].Contact\_Name\_\_c == '' ||

//Farmer.State\_\_c== '' ||

//farmersList[i].Edad\_\_c == '' ||

//farmersList[i].Genero\_\_c == '' ||

farmersList[i].TIPO\_DE\_INSUMO\_\_c == '' ||

farmersList[i].Sub\_Type\_\_c == '' ||

farmersList[i].Quantity\_\_c == '' )

{

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " +(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

//final method to create records in server side

saveFarmerList: function(component, event, helper) {

var FarmerId = component.get("v.farmerId");

console.log('farmerid>>'+ FarmerId);

// Adding Participant Gather Type in ContactList

var FarmerList = component.get("v.Farmer");

console.log('FarmerList>>'+ FarmerList);

//Call Apex class and pass list of parameters

var action = component.get("c.saveFarmer");

action.setParams({

"FarmerList": component.get("v.Farmer"),

});

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

component.set("v.Farmer", []);

// alert('Registros guardados ')

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action);

},

})

**Css:**

.THIS .slds-icon-utility-delete.slds-icon\_container{

padding-top: 14px;

padding-left: 18px;

}

.THIS .message .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .message .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"complete este campo";

}

.THIS .message .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public class AG\_Input\_Form1Controller {

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list< fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag input Form' limit 1];

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.ExtenAgent\_\_c = wrapper.ExtensionistaId;

far1.TIPO\_DE\_INSUMO\_\_c = wrapper.TipoDeInsumo;

far1.Sub\_Type\_\_c = wrapper.SubTipo;

far1.Quantity\_\_c = wrapper.Cantidad;

if(wrapper.Identificacion!= null){

far1.Code\_\_c = wrapper.Identificacion;

}

ListtoInsert.add(far1);

}

if(ListtoInsert.size() > 0){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String ExtensionistaId;

public String TipoDeInsumo;

public String SubTipo;

public String Cantidad;

public String Identificacion;

}

@AuraEnabled

public static void saveFarmer(List<Farmers\_\_c> FarmerList){

//Insert FarmerList;

system.debug('FarmerList>>'+FarmerList);

list<Farmers\_\_c> finalFarmerlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag input Form' limit 1];

for(Farmers\_\_c far : Farmerlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmerlist.add(far);

}

system.debug('finalFarmerlist>>'+finalFarmerlist);

Insert finalFarmerlist;

}

@AuraEnabled

public static Map<String, String> getTypeMethod(){

Map<String, String> optionsType = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.TIPO\_DE\_INSUMO\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsType.put(ple.getValue(), ple.getLabel());

}

return optionsType;

}

@AuraEnabled

public static Map<String, String> getAge(){

Map<String, String> optionsAge = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Edad\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsAge.put(ple.getValue(), ple.getLabel());

}

return optionsAge;

}

@AuraEnabled

public static Map<String, String> getGender(){

Map<String, String> optionsGender = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Genero\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsGender.put(ple.getValue(), ple.getLabel());

}

return optionsGender;

}

}

**TestClass:**

@isTest

public class AG\_Input\_Form1ControllerTest {

@istest

static void AgInputFormTest(){

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag input Form' limit 1];

system.debug('RT@@'+RT);

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

far1.TIPO\_DE\_INSUMO\_\_c='Semillas';

far1.Sub\_Type\_\_c ='test123';

far1.Quantity\_\_c='test';

far1.Contact\_Name\_\_c= con.id;

far1.RecordTypeId = RT.Id;

insert finalFarmlist1;

AG\_Input\_Form1Controller.saveFarmer(finalFarmlist1);

Map<String, String> getType = AG\_Input\_Form1Controller.getTypeMethod();

system.assertNotEquals(Null, getType);

Map<String, String> getAge = AG\_Input\_Form1Controller.getAge();

system.assertNotEquals(Null, getAge);

Map<String, String> getGender = AG\_Input\_Form1Controller.getGender();

system.assertNotEquals(Null, getGender);

}

@isTest

static void AgInputFormCSVTest(){

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag input Form' limit 1];

system.debug('RT@@'+RT);

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

far2.TIPO\_DE\_INSUMO\_\_c='Semillas';

far2.Sub\_Type\_\_c ='test123';

far2.Quantity\_\_c='test';

far2.Contact\_Name\_\_c= con.id;

far2.RecordTypeId = RT.Id;

finalFarmlist1.add(far2);

insert finalFarmlist1;

// AG\_Input\_Form1Controller.saveFarmer(finalFarmlist1);

AG\_Input\_Form1Controller.insertData(JSON.serialize(finalFarmlist1));

Map<String, String> getType = AG\_Input\_Form1Controller.getTypeMethod();

system.assertNotEquals(Null, getType);

Map<String, String> getAge = AG\_Input\_Form1Controller.getAge();

system.assertNotEquals(Null, getAge);

Map<String, String> getGender = AG\_Input\_Form1Controller.getGender();

system.assertNotEquals(Null, getGender);

}

}

**Ag Training cmp:**

<aura:component controller="Ag\_Training\_Participant" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!--Attribute declaration-->

<aura:attribute name="FarmerList" type="Farmers\_\_c[]"/>

<aura:attribute name="farm" type="Farmers\_\_c" default="{'sObjectType' : 'Farmers\_\_c'}"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!--Attribute declaration for picklist values-->

<aura:attribute name="TypeOfTrainerMap" type="Map"/>

<aura:attribute name="options\_TypeOfTrainer" type="List" default="[]"/>

<aura:attribute name="TypeOfTrainingMap" type="Map"/>

<aura:attribute name="options\_TypeOfTraining" type="List" default="[]"/>

<aura:attribute name="TypeOfGenderMap" type="Map"/>

<aura:attribute name="options\_TypeOfGender" type="List" default="[]"/>

<aura:attribute name="TypeOfTAFAMap" type="Map"/>

<aura:attribute name="options\_TypeOfTAFA" type="List" default="[]"/>

<aura:attribute name="TypeOfAgeMap" type="Map"/>

<aura:attribute name="options\_TypeOfAge" type="List" default="[]"/>

<!--Attribute declaration for lookup field-->

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<aura:handler name="change" value="{!v.selectedContactId}" action="{!c.handleContactChange}" />

<aura:attribute name="selectedContactId" type="String" />

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<!-- <aura:attribute name="trainerState" type="boolean" default="false"/>

<aura:attribute name="trainingName" type="boolean" default="false"/>

<aura:attribute name="trainerType" type="boolean" default="false"/>

<aura:attribute name="trainingType" type="boolean" default="false"/> -->

<!--Attribute declaration for text field -->

<aura:attribute name="NameOTG" type="String"/>

<aura:attribute name="FarmState" type="String"/>

<aura:attribute name="TOTrainer" type="String"/>

<aura:attribute name="TOTraining" type="String"/>

<aura:attribute name="TRDate" type="String"/>

<!-- Action method/doInit-->

<!--<aura:handler name="init" value="{!this}" action="{!c.doInit}"/>-->

<aura:handler name="init" value="{!this}" action="{!c.doTypeOfTraining}"/>

<aura:handler name="init" value="{!this}" action="{!c.doAssociation}"/>

<!-- <aura:handler name="init" value="{!this}" action="{!c.doTypeOfGender}"/>

<aura:handler name="init" value="{!this}" action="{!c.doAgetype}"/>-->

<!--lightning components for designing UI-->

<div class="slds-m-around--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!--<img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Arial; color:green;">Formulario de participantes de capacitación agrícola</span>

</div>

</div>

<lightning:layout>

<!-- <lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Estado<abbr class="slds-required" >\*</abbr></label>

<lightning:input label="State" value="{!v.State\_\_c}"

aura:id="training" required="true" onchange="{!c.captureState}" variant="label-hidden" class="message"/>

</div>

</lightning:layoutItem>-->

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Nombre de capacitación<abbr class="slds-required" >\*</abbr></label>

<lightning:input label="Name of training" value="{!v.Name\_of\_training\_\_c}"

aura:id="trainingName" required="true" onchange="{!c.captureNOTG}" variant="label-hidden" class="message"/>

</div>

</lightning:layoutItem>

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_small slds-p-bottom\_small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Fecha<abbr class="slds-required" >\*</abbr> </label>

<lightning:input type="text" name="input" pattern="^((0[1-9])|([12][0-9])|(3[01]))\/((0[1-9])|(1[0-2]))\/((19[0-9]{2})|(2[0-9]{3}))$" required="true" onchange="{!c.captureDate}" class="date" placeholder="DD/MM/AAAA" variant="label-hidden"/>

</div>

</lightning:layoutItem>

</lightning:layout>

<lightning:layout>

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_small slds-p-bottom\_small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Tipo de entrenador<abbr class="slds-required" >\*</abbr></label>

<!-- <lightning:combobox aura:id="trainer" label="Type Of Trainer" value="{!v.Tipo\_de\_entrenador\_\_c}" options="{!v.options\_TypeOfTrainer}" placeholder="" onchange="{!c.captureTOTP}" variant="label-hidden" class="stateFieldSize"/>-->

<lightning:input label="Type of Trainer" value="{!v.Type\_of\_Trainers\_\_c}"

aura:id="trainer" required="true" onchange="{!c.captureTOTP}" variant="label-hidden" class="message"/>

</div>

</lightning:layoutItem>

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_small slds-p-bottom\_small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Tipo de capacitación <abbr class="slds-required" >\*</abbr></label>

<lightning:combobox aura:id="trainerId" label="Type Of Training" required="true" value="{!v.Tipo\_de\_capacitaci\_n\_\_c}" options="{!v.options\_TypeOfTraining}" placeholder="" onchange="{!c.captureTOTG}" variant="label-hidden" class="message"/>

</div>

</lightning:layoutItem>

</lightning:layout>

<!-- <lightning:layout>

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_small slds-p-bottom\_small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Fecha<abbr class="slds-required" >\*</abbr> </label>

<lightning:input label="Date" type="date" required="true" onchange="{!c.captureDate}" variant="label-hidden" class="date" dateStyle="short" />

<lightning:input type="text" name="input" pattern="^((0[1-9])|([12][0-9])|(3[01]))\/((0[1-9])|(1[0-2]))\/((19[0-9]{2})|(2[0-9]{3}))$" required="true" onchange="{!c.captureDate}" class="date" placeholder="DD/MM/AAAA" variant="label-hidden"/>

<ui:inputDate label="" format="DD-MM-YYYY" displayDatePicker="true" />

<lightning:formattedNumber value="22-11-2021"/>

</div>

</lightning:layoutItem>

</lightning:layout>-->

</div>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-truncate">S.No</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="ID" align="center">Identificación<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th>

<div class="slds-cell-wrap" title="Age" align="center">Edad<abbr class="slds-required" >\*</abbr></div>

</th>

<th>

<div class="slds-cell-wrap" title="Gender" align="center">Genero<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<div class="slds-cell-wrap" title="Name of association or affiliation" align="center">Nombre de asociación <abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Type of association farmer is affiiliated with" align="center">Tipo de asociación a la que está afiliado el agricultor<abbr class="slds-required" >\*</abbr>

</div>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="Action">Action</div>

</th>-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.FarmerList}" var="Far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!-- <lightning:input name="FarID" type="text" />-->

<!--<c:LookupPersonId objectAPIName="contact" IconName="standard:contact" />-->

<!-- Calling another component for creating lookup -->

<c:ContactFarmerLookUp

objectAPIName="Contact"

label="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId ="{!Far.Contact\_Name\_\_c}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<!-- <td>

<lightning:combobox aura:id="Age" label="Age" value="{!Far.Edad\_\_c}" options="{!v.options\_TypeOfAge}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox aura:id="Gender" label="Gender" value="{!Far.Genero\_\_c}" options="{!v.options\_TypeOfGender}" placeholder="" variant="label-hidden"/>

</td>-->

<td>

<lightning:input name="Name of association or affiliation" type="Name of association" value="{!Far.Name\_of\_association\_or\_affiliation\_\_c}" variant="label-hidden"/>

</td>

<td>

<lightning:combobox aura:id="TAFA" label="Type of association" value="{!Far.Tipo\_de\_asociaci\_n\_a\_la\_que\_est\_afiliad\_\_c}" options="{!v.options\_TypeOfTAFA}" placeholder="" variant="label-hidden"/>

</td>

<!-- <td>

<a onclick="{!c.removeRow}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td> -->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 70rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario de participantes de capacitación agrícola</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

</aura:component>

**Controller:**

({

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("Nombredeasociacion", "Tipodeasociacionalaqueestaafiliadoelagricultor", "Nombredecapacitacion", "Fecha" , "Tipodeentrenador", "Tipodecapacitacion", "Identificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

//alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

// alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

// alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 7){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("Nombredeasociacion", "Tipodeasociacionalaqueestaafiliadoelagricultor", "Nombredecapacitacion", "Fecha" , "Tipodeentrenador", "Tipodecapacitacion", "Identificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

// alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 7){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

doTypeOfTraining : function(component, event, helper) {

helper.addFarmerRecord(component, event);

var action = component.get("c.TypeOfTrainingMethod");

// alert('Do type of Training');

action.setCallback(this, function(response) {

var state = response.getState();

// alert('trainingstate>>'+ state);

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeOfTrainingMap = [];

var TypeOfTrainingOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

TypeOfTrainingMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeOfTrainingOptions.push(state);

// alert(JSON.stringify(TypeOfTrainingOptions));

//system.debug('IndividualTypeOptions'+JSON.Stringfy(IndividualTypeOptions));

}

component.set("v.TypeOfTrainingMap", TypeOfTrainingMap);

component.set("v.options\_TypeOfTraining", TypeOfTrainingOptions);

}

});

$A.enqueueAction(action);

},

doTypeOfGender : function(component, event, helper) {

var action = component.get("c.getGendermapMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeOfGenderMap = [];

var TypeOfGenderOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

TypeOfGenderMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeOfGenderOptions.push(state);

}

component.set("v.TypeOfGenderMap", TypeOfGenderMap);

component.set("v.options\_TypeOfGender", TypeOfGenderOptions);

}

});

$A.enqueueAction(action);

},

doAgetype : function(component, event, helper) {

var action = component.get("c.getAgemapMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeOfAgeMap = [];

var TypeOfAgeOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

TypeOfAgeMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeOfAgeOptions.push(state);

}

component.set("v.TypeOfAgeMap", TypeOfAgeMap);

component.set("v.options\_TypeOfAge", TypeOfAgeOptions);

}

});

$A.enqueueAction(action);

},

doAssociation : function(component, event, helper) {

var action = component.get("c.getTAFAmapMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeOfTAFAMap = [];

var TypeOfTAFAOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

TypeOfTAFAMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeOfTAFAOptions.push(state);

}

component.set("v.TypeOfTAFAMap", TypeOfTAFAMap);

component.set("v.options\_TypeOfTAFA", TypeOfTAFAOptions);

}

});

$A.enqueueAction(action);

},

//Select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.FarmerList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//add new row method

addRow: function(component, event, helper) {

helper.addFarmerRecord(component, event);

},

//remove row method(single row)

removeRow: function(component, event, helper) {

//Get the Farmer list

var FarmerList = component.get("v.FarmerList");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

FarmerList.splice(index, 1);

component.set("v.FarmerList", FarmerList);

},

//final method to create records in server side

save: function(component, event, helper) {

var FarmerList = component.get("v.FarmerList");

//Validating fields TOTraining,TOTrainer etc...

var TOTGValue = component.get("v.TOTraining");

var TOTRValue = component.get("v.TOTrainer");

var StateValue = component.get("v.FarmState");

var NOTGValue = component.get("v.NameOTG");

var DateValue = component.get("v.TRDate");

// if(typeof TOTGValue == 'undefined' || typeof TOTRValue == 'undefined' ||

//typeof StateValue == 'undefined' || typeof NOTGValue == 'undefined' ||

// typeof DateValue == 'undefined'){

if(typeof TOTGValue == 'undefined' || typeof TOTRValue == 'undefined' ||

typeof NOTGValue == 'undefined' ||

typeof DateValue == 'undefined'){

//alert('Las celdas obligatorias no pueden estar en blanco');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

else{

if (helper.validateFarmerRecords(component, event)) {

var TOTGValue = component.get("v.TOTraining");

var TOTRValue = component.get("v.TOTrainer");

var StateValue = component.get("v.FarmState");

var NOTGValue = component.get("v.NameOTG");

var DateValue = component.get("v.TRDate");

// Adding Type of Trainer and Trainer all values adding to FarmerList

var FarmerList = component.get("v.FarmerList");

for (var i = 0; i < FarmerList.length; i++) {

FarmerList[i].Tipo\_de\_capacitaci\_n\_\_c = TOTGValue;

FarmerList[i].Type\_of\_Trainers\_\_c = TOTRValue;

// FarmerList[i].State\_\_c = StateValue;

FarmerList[i].Name\_of\_training\_\_c = NOTGValue;

FarmerList[i].Training\_date\_\_c = DateValue;

}

////Call Apex class and pass Farmer list parameters

var action = component.get("c.saveAgTraining");

action.setParams({

"Farlist": component.get("v.FarmerList")

});

//console.log(action);

action.setCallback(this, function(response) {

var state = response.getState();

//alert('state@@'+state);

if (state === "SUCCESS") {

component.set("v.FarmerList", []);

//alert('Registros guardados '); //eng translation for alert:Records Saved Successfully

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action);

}

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

//getting values from client side

captureTOTG :function(component,event){

var TOTGValue = event.getParam("value");

//alert(TOTGValue);

component.set("v.TOTraining", TOTGValue);

},

captureTOTP :function(component,event){

var TOTRValue = event.getParam("value");

//alert(TOTRValue);

component.set("v.TOTrainer", TOTRValue);

},

captureState :function(component,event){

var StateValue = event.getParam("value");

//alert(StateValue);

component.set("v.FarmState", StateValue);

},

captureNOTG :function(component,event){

var NOTGValue = event.getParam("value");

//alert(NOTGValue);

component.set("v.NameOTG", NOTGValue);

},

captureDate :function(component,event){

var TGDateValue = event.getParam("value");

// alert(TGDateValue);

component.set("v.TRDate", TGDateValue);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

readCSV: function(component, event, helper) {

const headers2 = new Array("Nombredeasociacion", "Tipodeasociacionalaqueestaafiliadoelagricultor", "Nombredecapacitacion", "Fecha" , "Tipodeentrenador", "Tipodecapacitacion", "Identificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 7){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].ExtensionistaId=="" ||

data1[k].TipoDeInsumo==""||

data1[k].SubTipo=="" ||

data1[k].Cantidad=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var Nombredeasociacion;

var Tipodeasociacionalaqueestaafiliadoelagricultor;

var Nombredecapacitacion;

var Fecha;

var Tipodeentrenador;

var Tipodecapacitacion;

var Identificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

// alert(rows);

arr.pop();

// alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 7){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].Nombredeasociacion=="" ||

data1[k].Tipodeasociacionalaqueestaafiliadoelagricultor==""||

data1[k].Nombredecapacitacion=="" ||

data1[k].Fecha=="" ||

data1[k].Tipodeentrenador=="" ||

data1[k].Tipodecapacitacion=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var Nombredeasociacion;

var Tipodeasociacionalaqueestaafiliadoelagricultor;

var Nombredecapacitacion;

var Fecha;

var Tipodeentrenador;

var Tipodecapacitacion;

var Identificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

CSV2JSON: function (component,csv) {

// console.log('Incoming csv = ' + csv);

//var array = [];

var arr = [];

arr = csv.split('\n');

//console.log('Array = '+array);

// console.log('arr = '+arr);

arr.pop();

var jsonObj = [];

var headers = arr[0].split(',');

for(var i = 1; i < arr.length; i++) {

var data = arr[i].split(',');

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

jsonObj.push(obj);

}

var data1 = jsonObj;

for( var k=0;k<jsonObj.length;k++){

//data1 = JSON.stringify(jsonObj[k]);

//data1 = jsonObj[k];

// data2 = JSON.stringify(data1)

console.log(data1);

if(data1[k].Nombredeasociacion==""||

data1[k].Tipodeasociacionalaqueestaafiliadoelagricultor=="" ||

data1[k].Nombredecapacitacion==""||

data1[k].Fecha=="" ||

data1[k].Tipodeentrenador=="" ||

data1[k].Tipodecapacitacion=="" ||

data1[k].Identificacion==""){

//alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var json = JSON.stringify(data1);

console.log('json = '+ json);

return json;

},

CreateContact : function (component,jsonstr){

// console.log('jsonstr' + jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

//alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

// alert('result callback>>'+ result);

// alert("Registros guardados");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

addFarmerRecord: function(component, event) {

//get the Farmer List from component

var FarmerList = component.get("v.FarmerList");

//Add New Farmer Record

FarmerList.push({

'sobjectType': 'Farmers\_\_c',

'State\_\_c': '',

'Tipo\_de\_entrenador\_\_c': '',

'Tipo\_de\_capacitaci\_n\_\_c':'',

'Name\_of\_training\_\_c' : '',

'Training\_date\_\_c' : '',

'Name\_of\_association\_or\_affiliation\_\_c':'',

'Tipo\_de\_asociaci\_n\_a\_la\_que\_est\_afiliad\_\_c':'',

'Edad\_\_c':'',

'Genero\_\_c':'',

'Contact\_Name\_\_c' : ''

});

component.set("v.FarmerList", FarmerList);

},

//checking validation for all required fields

validateFarmerRecords: function(component, event) {

//Validate all Farmer records

var isValid = true;

var FarmerList = component.get("v.FarmerList");

for (var i = 0; i < FarmerList.length; i++) {

if (FarmerList[i].Contact\_Name\_\_c == '' ||

// FarmerList[i].Edad\_\_c == '' ||

// FarmerList[i].Genero\_\_c == '' ||

FarmerList[i].Name\_of\_association\_or\_affiliation\_\_c == '' ||

FarmerList[i].Tipo\_de\_asociaci\_n\_a\_la\_que\_est\_afiliad\_\_c == '' )

{

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " +(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

//Process the selected Farmers records for remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.FarmerList");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.FarmerList", finalFarmers);

},

})

**Css:**

.THIS .slds-icon-utility-delete.slds-icon\_container{

padding-top: 14px;

padding-left: 18px;

}

.THIS .date .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .date .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"Utilice el siguiente formato: DD/MM/AAAA";

}

.THIS .date .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .message .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .message .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"complete este campo";

}

.THIS .message .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public class Ag\_Training\_Participant {

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list< fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag Training Participant Form' limit 1];

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.Name\_of\_association\_or\_affiliation\_\_c = wrapper.Nombredeasociacion;

far1.Tipo\_de\_asociaci\_n\_a\_la\_que\_est\_afiliad\_\_c = wrapper.Tipodeasociacionalaqueestaafiliadoelagricultor;

far1.Name\_of\_training\_\_c = wrapper.Nombredecapacitacion;

far1.Training\_date\_\_c = wrapper.Fecha;

far1.Type\_of\_Trainers\_\_c = wrapper.Tipodeentrenador;

far1.Tipo\_de\_capacitaci\_n\_\_c = wrapper.Tipodecapacitacion;

if(wrapper.Identificacion!= null){

far1.Code\_\_c = wrapper.Identificacion;

}

ListtoInsert.add(far1);

}

if(ListtoInsert.size() > 0){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String Nombredeasociacion;

public String Tipodeasociacionalaqueestaafiliadoelagricultor;

public String Nombredecapacitacion;

public String Fecha;

public String Tipodeentrenador;

public String Tipodecapacitacion;

public String Identificacion;

}

@AuraEnabled

public static void saveAgTraining(list<Farmers\_\_c> Farlist){

//Insert Farlist;

//system.debug('Farmlist>>'+Farmlist);

list<Farmers\_\_c> finalFarmlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag Training Participant Form' limit 1];

for(Farmers\_\_c far : Farlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmlist.add(far);

}

system.debug('finalFarmlist>>'+finalFarmlist);

Insert finalFarmlist;

}

@AuraEnabled

public static Map<String, String> TypeOfTrainerMethod(){

Map<String, String> optionsTypeOfTrainer = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Tipo\_de\_entrenador\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsTypeOfTrainer.put(ple.getValue(), ple.getLabel());

}

return optionsTypeOfTrainer;

}

@AuraEnabled

public static Map<String, String> TypeOfTrainingMethod(){

Map<String, String> optionsTypeOfTraining = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Tipo\_de\_capacitaci\_n\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsTypeOfTraining.put(ple.getValue(), ple.getLabel());

}

return optionsTypeOfTraining;

}

@AuraEnabled

public static Map<String, String> getGendermapMethod(){

Map<String, String> optionsOfGendermap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Genero\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsOfGendermap.put(ple.getValue(), ple.getLabel());

}

return optionsOfGendermap;

}

@AuraEnabled

public static Map<String, String> getAgemapMethod(){

Map<String, String> optionsAgemap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Edad\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsAgemap.put(ple.getValue(), ple.getLabel());

}

return optionsAgemap;

}

@AuraEnabled

public static Map<String, String> getTAFAmapMethod(){

Map<String, String> optionsTAFAmap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Tipo\_de\_asociaci\_n\_a\_la\_que\_est\_afiliad\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsTAFAmap.put(ple.getValue(), ple.getLabel());

}

return optionsTAFAmap;

}

}

**TestClass:**

@isTest

public class Ag\_Training\_ParticipantTest {

@istest

static void AgTrainingParticipantTest(){

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag Training Participant Form' limit 1];

system.debug('RT@@'+RT);

Ag\_Training\_Participant ag = new Ag\_Training\_Participant();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

far1.State\_\_c='TamilNadu';

far1.Name\_of\_training\_\_c ='test123';

far1.Training\_date\_\_c='11/12/2022';

far1.Tipo\_de\_capacitaci\_n\_\_c='producción';

far1.Tipo\_de\_asociaci\_n\_a\_la\_que\_est\_afiliad\_\_c='instituciones educativas';

far1.Genero\_\_c='Mujer';

far1.Edad\_\_c='30 y más';

far1.Contact\_Name\_\_c= con.id;

far1.RecordTypeId = RT.Id;

insert finalFarmlist1;

Ag\_Training\_Participant.saveAgTraining(finalFarmlist1);

Map<String, String> typeofTrainer = Ag\_Training\_Participant.TypeOfTrainerMethod();

system.assertNotEquals(Null, typeofTrainer);

Map<String, String> typeofTraining = Ag\_Training\_Participant.TypeOfTrainingMethod();

system.assertNotEquals(Null, typeofTraining);

Map<String, String> GenderMap = Ag\_Training\_Participant.getGendermapMethod();

system.assertNotEquals(Null, GenderMap);

Map<String, String> AgeMap = Ag\_Training\_Participant.getAgemapMethod();

system.assertNotEquals(Null, AgeMap);

Map<String, String> TFTAmap = Ag\_Training\_Participant.getTAFAmapMethod();

system.assertNotEquals(Null, TFTAmap);

}

@istest

static void AgTrainingParticipantCSVTest(){

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag Training Participant Form' limit 1];

system.debug('RT@@'+RT);

Ag\_Training\_Participant ag = new Ag\_Training\_Participant();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

far2.State\_\_c='TamilNadu';

far2.Name\_of\_training\_\_c ='test123';

far2.Training\_date\_\_c='11/12/2022';

far2.Tipo\_de\_capacitaci\_n\_\_c='producción';

far2.Tipo\_de\_asociaci\_n\_a\_la\_que\_est\_afiliad\_\_c='instituciones educativas';

far2.Genero\_\_c='Mujer';

far2.Edad\_\_c='30 y más';

far2.Contact\_Name\_\_c= con.id;

far2.RecordTypeId = RT.Id;

finalFarmlist1.add(far2);

insert finalFarmlist1;

//Ag\_Training\_Participant.saveAgTraining(finalFarmlist1);

Ag\_Training\_Participant.insertData(JSON.serialize(finalFarmlist1));

Map<String, String> typeofTrainer = Ag\_Training\_Participant.TypeOfTrainerMethod();

system.assertNotEquals(Null, typeofTrainer);

Map<String, String> typeofTraining = Ag\_Training\_Participant.TypeOfTrainingMethod();

system.assertNotEquals(Null, typeofTraining);

Map<String, String> GenderMap = Ag\_Training\_Participant.getGendermapMethod();

system.assertNotEquals(Null, GenderMap);

Map<String, String> AgeMap = Ag\_Training\_Participant.getAgemapMethod();

system.assertNotEquals(Null, AgeMap);

Map<String, String> TFTAmap = Ag\_Training\_Participant.getTAFAmapMethod();

system.assertNotEquals(Null, TFTAmap);

}

}

**Ag Post Training cmp:**

<aura:component controller="AG\_Post\_TrainingController" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Aura Declaration-->

<aura:attribute name="FarmerList" type="Farmers\_\_c[]"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!--Attribute declaration for picklistlist values-->

<aura:attribute name="TechniquesMap" type="Map"/>

<aura:attribute name="options\_TechniquesMap" type="List" default="[]"/>

<aura:attribute name="TypeofhectareMap" type="Map"/>

<aura:attribute name="options\_TypeofhectareMap" type="List" default="[]"/>

<aura:attribute name="TechTypeMap" type="Map"/>

<aura:attribute name="options\_TechTypeMap" type="List" default="[]"/>

<aura:attribute name="NewTechniquesMap" type="Map"/>

<aura:attribute name="options\_NewTechniquesMap" type="List" default="[]"/>

<aura:attribute name="CommodityMap" type="Map"/>

<aura:attribute name="options\_commodityMap" type="List" default="[]"/>

<aura:attribute name="trainerState" type="boolean" default="false"/>

<aura:attribute name="trainingDate" type="boolean" default="false"/>

<aura:attribute name="GenderMap" type="Map"/>

<aura:attribute name="options\_GenderMap" type="List" default="[]"/>

<aura:attribute name="TypeofAgeMap" type="Map"/>

<aura:attribute name="options\_TypeofAgeMap" type="List" default="[]"/>

<!--Attribute declaration for lookup field-->

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<!--Action Method/doInit-->

<aura:handler name="init" value="{!this}" action="{!c.doInit}"/>

<aura:handler name="init" value="{!this}" action="{!c.doTypeofhectare}"/>

<aura:handler name="init" value="{!this}" action="{!c.doTechTypeMap}"/>

<aura:handler name="init" value="{!this}" action="{!c.doNewTechnique}"/>

<aura:handler name="init" value="{!this}" action="{!c.doCommodityType}"/>

<!-- <aura:handler name="init" value="{!this}" action="{!c.doGender}"/>

<aura:handler name="init" value="{!this}" action="{!c.doAge}"/>-->

<aura:html tag="style">

.slds-col--padded.contentRegion.comm-layout-column{

width:1300px !important;

margin-left:-80px !important;

}

</aura:html>

<!--Lightning component for designing UI-->

<div class="slds-m-around--xxx--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!-- <img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Times New Roman; color:green;">Formulario de seguimiento de la capacitación agrícola </span>

</div>

<lightning:layout>

<!-- <lightning:layoutitem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small ">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Estado<abbr class="slds-required" >\*</abbr></label>

<lightning:input label="State" aura:id="training" value="{!v.trainerState}" required="true" variant="label-hidden" />

</div>

</lightning:layoutitem>-->

<lightning:layoutitem size="6" >

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small ">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Fecha de visita de seguimento con agricultor<abbr class="slds-required" >\*</abbr></label>

<!-- <lightning:input type="Date" label="Date of Follow Up Visit To Farmer" aura:id="trainingDateId" value="{!v.trainingDate}" required="true" variant="label-hidden" dateStyle="short"/>-->

<lightning:input type="text" name="input" pattern="^((0[1-9])|([12][0-9])|(3[01]))\/((0[1-9])|(1[0-2]))\/((19[0-9]{2})|(2[0-9]{3}))$" required="true" value="{!v.trainingDate}" class="date" placeholder="DD/MM/AAAA" variant="label-hidden"/>

</div>

</lightning:layoutitem>

</lightning:layout>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<!-- <input type="file" class="file" aura:id="file" onchange="{!c.showfiledata}"/>-->

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-truncate" >S.No</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="ID" align="center" >Identificación<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-cell-wrap" title="Age" align="center" >Edad<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Gender" align="center" >Genero<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<div class="slds-cell-wrap" title="Commodity" align="center">Producto<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Applying new technique " align="center">Usando nueva practica <abbr class="slds-required" >\*</abbr>

<!-- <lightning:helptext content="Yes Or No" class="customIcon"/>-->

</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Name of new techniques applied" align="center">Nombre de nueva practica utilizada</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Corresponding number on list of techniques" align="center">Número correspondiente en la lista de técnicas </div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Type of hectare " align="center">Tipo de hectárea<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Management Practice/Tech Type " align="center">Práctica de gestión / Tipo de técnica<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="No.of Hectares " align="center">Número de hectáreas usando nuevas prácticas<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="Action" align="center">Action</div>

</th>-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.FarmerList}" var="Far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!-- <lightning:input name="ID" type="text" value="{!Far.ID\_\_c}" />-->

<!--<c:LookupPersonId objectAPIName="contact" IconName="standard:contact" />-->

<!-- Calling another component for creating lookup -->

<c:ContactFarmerLookUp

objectAPIName="Contact"

label="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId ="{!Far.Contact\_Name\_\_c}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<!-- <td>

<lightning:combobox name="Age" type="number" value="{!Far.Edad\_\_c}" options="{!v.options\_TypeofAgeMap}" placeholder="" variant="label-hidden" />

</td>

<td>

<lightning:combobox name="Gender" type="text" value="{!Far.Genero\_\_c}" options="{!v.options\_GenderMap}" placeholder="" variant="label-hidden"/>

</td>-->

<td>

<lightning:combobox name="Commodity" type="Commodity" value="{!Far.NOMBRE\_DE\_PRODUCTO\_\_c}" options="{!v.options\_commodityMap}" variant="label-hidden" placeholder=""/>

</td>

<td>

<lightning:combobox name="Applying new technique " type="Technique" value="{!Far.USANDO\_NUEVA\_PRACTICA\_\_c}" options="{!v.options\_NewTechniquesMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:input name="Name of new techniques applied" type="text" value="{!Far.Name\_of\_new\_techniques\_applied\_\_c}" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="Corresponding number on list of techniques" type="Technique" value="{!Far.N\_mero\_correspondiente\_en\_la\_lista\_de\_t\_\_c}" options="{!v.options\_TechniquesMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="Type of hectare" type="Type of hectare" value="{!Far.Tipo\_de\_hect\_rea\_\_c}" options="{!v.options\_TypeofhectareMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="Management Practice/Tech Type" type="Management Practice/Tech Type" value="{!Far.Pr\_ctica\_de\_gesti\_n\_Tipo\_de\_t\_cnica\_\_c}" options="{!v.options\_TechTypeMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:input name="No.of Hectares" type="Number" maxlength="10" value="{!Far.No\_of\_hectares\_\_c}" variant="label-hidden"/>

</td>

<!-- <td>

<a onclick="{!c.removeRow}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td> -->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 90rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario de seguimiento de la capacitación agrícola</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

</aura:component>

**Controller:**

({

// Showing the values in Popup while uploading files in both .csv and .xlsx format

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("FechadeVisita", "Producto", "Usandonuevapractica", "Nombredenuevapracticautilizada", "Numerocorrespondienteenlalistadetecnicas", "Tipodehectarea", "Practicadegestion", "Numerodehectareasusandonuevaspracticas", "Identificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

// alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

//alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

// alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("FechadeVisita", "Producto", "Usandonuevapractica", "Nombredenuevapracticautilizada", "Numerocorrespondienteenlalistadetecnicas", "Tipodehectarea", "Practicadegestion", "Numerodehectareasusandonuevaspracticas", "Identificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

//alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

//Calling serverside/apex controller for getting Picklist Values

doInit : function(component, event, helper) {

/\* document.onkeydown = function(evt) {

if(evt.key == "Escape"){

component.set("v.closeModal", false);

// location.reload();

}

}\*/

helper.addFarmerRecord(component, event);

var action = component.get("c.getTechniques");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TechniquesMap = [];

var TechniquesMapOptions = [];

for(var key in result){

TechniquesMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TechniquesMapOptions.push(state);

//console.log(TechniquesMapOptions);

//system.debug('IndividualTypeOptions'+JSON.Stringfy(TechniquesMapOptions));

}

component.set("v.TechniquesMap", TechniquesMap);

component.set("v.options\_TechniquesMap", TechniquesMapOptions);

}

});

$A.enqueueAction(action);

},

doTypeofhectare : function(component, event, helper) {

var action = component.get("c.getTypeofhectare");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeofhectareMap = [];

var TypeofhectareMapOptions = [];

for(var key in result){

TypeofhectareMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeofhectareMapOptions.push(state);

}

component.set("v.TypeofhectareMap", TypeofhectareMap);

component.set("v.options\_TypeofhectareMap", TypeofhectareMapOptions);

}

});

$A.enqueueAction(action);

},

doTechTypeMap : function(component, event, helper) {

var action = component.get("c.getTechType");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TechTypeMap = [];

var TechTypeMapOptions = [];

for(var key in result){

TechTypeMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TechTypeMapOptions.push(state);

}

component.set("v.TechTypeMap", TechTypeMap);

component.set("v.options\_TechTypeMap", TechTypeMapOptions);

}

});

$A.enqueueAction(action);

},

doNewTechnique : function(component, event, helper) {

var action = component.get("c.getNewTechType");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var NewTechniquesMap = [];

var NewTechniquesMapOptions = [];

for(var key in result){

NewTechniquesMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

NewTechniquesMapOptions.push(state);

}

component.set("v.NewTechniquesMap", NewTechniquesMap);

component.set("v.options\_NewTechniquesMap", NewTechniquesMapOptions);

}

});

$A.enqueueAction(action);

},

doCommodityType : function(component, event, helper) {

var action = component.get("c.getCommodityType");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var CommodityMap = [];

var CommodityMapOptions = [];

for(var key in result){

CommodityMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

CommodityMapOptions.push(state);

}

component.set("v.CommodityMap", CommodityMap);

component.set("v.options\_commodityMap", CommodityMapOptions);

}

});

$A.enqueueAction(action);

},

//Select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.FarmerList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//adding new row method

addRow: function(component, event, helper) {

helper.addFarmerRecord(component, event);

},

// remove row method(single record/row)

removeRow: function(component, event, helper) {

//Get the Farmer list

var FarmerList = component.get("v.FarmerList");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

FarmerList.splice(index, 1);

component.set("v.FarmerList", FarmerList);

},

//final method to create records in server side

save: function(component, event, helper) {

var FarmerList = component.get("v.FarmerList");

//var trainerState = component.get("v.trainerState");

var TrainingDate = component.get("v.trainingDate");

//checking Validation for required fields trainerstate and trainer date

//if(trainerState == false || TrainingDate == false){

if(TrainingDate == false){

// alert('Las celdas obligatorias no pueden estar en blanco');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

else{

if (helper.validateFarmerRecords(component, event)) {

for(var i=0; i<FarmerList.length; i++){

// FarmerList[i].State\_\_c = component.get("v.trainerState");

FarmerList[i].Date\_of\_follow\_up\_visit\_to\_farmer\_\_c = component.get("v.trainingDate");

//alert('Please fill the State and Date Field');

}

// helper.saveFarmerList(component, event);

//Call Apex class and pass Farmer list parameters

var action = component.get("c.saveAgPostTraining");

action.setParams({

"Farmlist": component.get("v.FarmerList"),

});

console.log('Action>>'+ action);

action.setCallback(this, function(response) {

var state = response.getState();

console.log('State>>' + state)

if (state === "SUCCESS") {

component.set("v.FarmerList", []);

// alert('Registros guardados'); //eng translation for alert:Records Saved Successfully

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '20000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action);

}

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

// Saving the records while uploading both .csv and .xlsx format

readCSV: function(component, event, helper) {

const headers2 = new Array("FechadeVisita", "Producto", "Usandonuevapractica", "Nombredenuevapracticautilizada", "Numerocorrespondienteenlalistadetecnicas", "Tipodehectarea", "Practicadegestion", "Numerodehectareasusandonuevaspracticas", "Identificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].FechadeVisita=="" ||

data1[k].Producto==""||

data1[k].Usandonuevapractica=="" ||

// data1[k].Nombredenuevapracticautilizada=="" ||

// data1[k].Numerocorrespondienteenlalistadetecnicas=="" ||

data1[k].Tipodehectarea=="" ||

data1[k].Practicadegestion=="" ||

data1[k].Numerodehectareasusandonuevaspracticas=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

else if(data1[k].Usandonuevapractica =="sí" && data1[k].Nombredenuevapracticautilizada == "" || data1[k].Numerocorrespondienteenlalistadetecnicas=="" ){

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor complete el Nombre de nueva practica utilizada y Número correspondiente en la lista de técnicas " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var FechadeVisita ;

var Producto;

var Usandonuevapractica;

var Nombredenuevapracticautilizada;

var Numerocorrespondienteenlalistadetecnicas;

var Tipodehectarea;

var Practicadegestion;

var Numerodehectareasusandonuevaspracticas;

var Identificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

// alert(rows);

arr.pop();

// alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].FechadeVisita=="" ||

data1[k].Producto==""||

data1[k].Usandonuevapractica=="" ||

// data1[k].Nombredenuevapracticautilizada=="" ||

// data1[k].Numerocorrespondienteenlalistadetecnicas=="" ||

data1[k].Tipodehectarea=="" ||

data1[k].Practicadegestion=="" ||

data1[k].Numerodehectareasusandonuevaspracticas=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

else if(data1[k].Usandonuevapractica =="sí" && data1[k].Nombredenuevapracticautilizada == "" || data1[k].Numerocorrespondienteenlalistadetecnicas=="" ){

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor complete el Nombre de nueva practica utilizada y Número correspondiente en la lista de técnicas " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var FechadeVisita ;

var Producto;

var Usandonuevapractica;

var Nombredenuevapracticautilizada;

var Numerocorrespondienteenlalistadetecnicas;

var Tipodehectarea;

var Practicadegestion;

var Numerodehectareasusandonuevaspracticas;

var Identificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

CreateContact : function (component,jsonstr){

//console.log('jsonstr' + jsonstr);

//alert('jsonstr' + jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

// alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

//alert('result callback>>'+ result);

// alert("Registros guardados");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

addFarmerRecord: function(component, event) {

//get the Farmer List from component

var FarmerList = component.get("v.FarmerList");

//Add New Farmer Record

FarmerList.push({

'sobjectType': 'Farmers\_\_c',

//'State\_\_c': '',

'USANDO\_NUEVA\_PRACTICA\_\_c':'',

'N\_mero\_correspondiente\_en\_la\_lista\_de\_t\_\_c':'',

'Tipo\_de\_hect\_rea\_\_c' : '',

'Pr\_ctica\_de\_gesti\_n\_Tipo\_de\_t\_cnica\_\_c' : '',

'NOMBRE\_DE\_PRODUCTO\_\_c' : '' ,

'Name\_of\_new\_techniques\_applied\_\_c' : '',

'No\_of\_hectares\_\_c' : '',

'Date\_of\_follow\_up\_visit\_to\_farmer\_\_c' :'',

'Contact\_Name\_\_c' : '',

});

component.set("v.FarmerList", FarmerList);

},

//checking Validation for all required fields

validateFarmerRecords: function(component, event) {

//Validate all account records

var isValid = true;

var FarmerList = component.get("v.FarmerList");

for (var i = 0; i < FarmerList.length; i++) {

/\*alert('Applying\_new\_technique\_\_c>>'+ FarmerList[i].Applying\_new\_technique\_\_c);

alert('No\_of\_hectares\_\_c>>'+ FarmerList[i].No\_of\_hectares\_\_c);\*/

if (FarmerList[i].Contact\_Name\_\_c == '' ||

// FarmerList[i].Edad\_\_c == '' ||

//FarmerList[i].Genero\_\_c == '' ||

// //FarmerList[i].N\_mero\_correspondiente\_en\_la\_lista\_de\_t\_\_c == '' ||

FarmerList[i].USANDO\_NUEVA\_PRACTICA\_\_c == '' ||

FarmerList[i].Tipo\_de\_hect\_rea\_\_c == '' ||

FarmerList[i].Pr\_ctica\_de\_gesti\_n\_Tipo\_de\_t\_cnica\_\_c == '' ||

FarmerList[i].NOMBRE\_DE\_PRODUCTO\_\_c == '' ||

// FarmerList[i].Name\_of\_new\_techniques\_applied\_\_c == ''||

FarmerList[i].No\_of\_hectares\_\_c == '')

{

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila "+(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

else if (FarmerList[i].USANDO\_NUEVA\_PRACTICA\_\_c == 'sí' && ($A.util.isEmpty(FarmerList[i].Name\_of\_new\_techniques\_applied\_\_c) ||$A.util.isEmpty(FarmerList[i].N\_mero\_correspondiente\_en\_la\_lista\_de\_t\_\_c)) ) {

isValid = false;

//alert('por favor complete el Nombre de nueva practica utilizada y Número correspondiente en la lista de técnicas');// please fill the livestock details

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor complete el Nombre de nueva practica utilizada y Número correspondiente en la lista de técnicas",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

//Process the selected Farmers records to remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.FarmerList");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.FarmerList", finalFarmers);

},

})

**Css:**

.THIS .slds-icon-utility-delete.slds-icon\_container{

padding-top: 14px;

padding-left: 18px;

}

.THIS .slds-container--xxx--large.slds-box.slds-theme\_default{

width:1200px;

}

.THIS .date .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .date .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"Utilice el siguiente formato: DD/MM/AAAA";

}

.THIS .date .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public without sharing class AG\_Post\_TrainingController {

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list< fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag Post Training Follow up Form' limit 1];

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.Date\_of\_follow\_up\_visit\_to\_farmer\_\_c = wrapper.FechadeVisita;

far1.NOMBRE\_DE\_PRODUCTO\_\_c = wrapper.Producto;

far1.USANDO\_NUEVA\_PRACTICA\_\_c = wrapper.Usandonuevapractica;

far1.Name\_of\_new\_techniques\_applied\_\_c = wrapper.Nombredenuevapracticautilizada;

far1.N\_mero\_correspondiente\_en\_la\_lista\_de\_t\_\_c = wrapper.Numerocorrespondienteenlalistadetecnicas;

far1.Tipo\_de\_hect\_rea\_\_c = wrapper.Tipodehectarea;

far1.Pr\_ctica\_de\_gesti\_n\_Tipo\_de\_t\_cnica\_\_c = wrapper.Practicadegestion;

far1.No\_of\_hectares\_\_c = wrapper.Numerodehectareasusandonuevaspracticas;

far1.Code\_\_c = wrapper.Identificacion;

ListtoInsert.add(far1);

}

if(ListtoInsert.size() > 0 ){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String FechadeVisita;

public String Producto;

public String Usandonuevapractica;

public String Nombredenuevapracticautilizada;

public String Numerocorrespondienteenlalistadetecnicas;

public String Tipodehectarea;

public String Practicadegestion;

public Decimal Numerodehectareasusandonuevaspracticas;

public String Identificacion;

}

@AuraEnabled

public static void saveAgPostTraining(list<Farmers\_\_c> Farmlist){

//Upsert Conlist;

// Insert Farmlist;

system.debug('Farmlist>>'+Farmlist);

list<Farmers\_\_c> finalFarmlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag Post Training Follow up Form' limit 1];

for(Farmers\_\_c far : Farmlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmlist.add(far);

}

system.debug('finalFarmlist>>'+finalFarmlist);

Insert finalFarmlist;

/\* try{

Insert Farmlist;

return 'SUCCESS';

} catch(Exception e) {

System.debug('The following exception has occurred: ' +e.getLineNumber()+ e.getMessage());

return e.getMessage();

}\*/

}

@AuraEnabled

public static Map<String, String> getTechniques(){

Map<String, String> optionsTechniquesMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.N\_mero\_correspondiente\_en\_la\_lista\_de\_t\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsTechniquesMap.put(ple.getValue(), ple.getLabel());

}

return optionsTechniquesMap;

}

@AuraEnabled

public static Map<String, String> getTypeofhectare(){

Map<String, String> optionsTypeofhectareMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Tipo\_de\_hect\_rea\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsTypeofhectareMap.put(ple.getValue(), ple.getLabel());

}

return optionsTypeofhectareMap;

}

@AuraEnabled

public static Map<String, String> getTechType(){

Map<String, String> optionsTechTypeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Pr\_ctica\_de\_gesti\_n\_Tipo\_de\_t\_cnica\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsTechTypeMap.put(ple.getValue(), ple.getLabel());

}

return optionsTechTypeMap;

}

@AuraEnabled

public static Map<String, String> getNewTechType(){

Map<String, String> optionsNewTechTypeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.USANDO\_NUEVA\_PRACTICA\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsNewTechTypeMap.put(ple.getValue(), ple.getLabel());

}

return optionsNewTechTypeMap;

}

@AuraEnabled

public static Map<String, String> getCommodityType(){

Map<String, String> optionsCommodityMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.NOMBRE\_DE\_PRODUCTO\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsCommodityMap.put(ple.getValue(), ple.getLabel());

}

return optionsCommodityMap;

}

}

**TestClass:**

@istest

public class AG\_Post\_TrainingControllerTest {

@istest

static void saveAgPostTrainingTest(){

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag Post Training Follow up Form' limit 1];

system.debug('RT@@'+RT);

AG\_Post\_TrainingController agpost = new AG\_Post\_TrainingController();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

far1.USANDO\_NUEVA\_PRACTICA\_\_c='No';

far1.N\_mero\_correspondiente\_en\_la\_lista\_de\_t\_\_c='Genética de cultivos';

far1.Tipo\_de\_hect\_rea\_\_c='Tierra de cultivo';

far1.Pr\_ctica\_de\_gesti\_n\_Tipo\_de\_t\_cnica\_\_c='Cultivo genética';

far1.NOMBRE\_DE\_PRODUCTO\_\_c='Aguacate';

far1.Name\_of\_new\_techniques\_applied\_\_c='testing';

far1.No\_of\_hectares\_\_c=decimal.valueof(56);

far1.Date\_of\_follow\_up\_visit\_to\_farmer\_\_c='22/09/2021';

far1.Contact\_Name\_\_c=con.id;

far1.RecordTypeId = RT.Id;

insert finalFarmlist1;

AG\_Post\_TrainingController.saveAgPostTraining(finalFarmlist1);

Map<String, String> Techniques = AG\_Post\_TrainingController.getTechniques();

system.assertNotEquals(Null, Techniques);

Map<String, String> typeofHectare= AG\_Post\_TrainingController.getTypeofhectare();

system.assertNotEquals(Null, typeofHectare);

Map<String, String> techType = AG\_Post\_TrainingController.getTechType();

system.assertNotEquals(Null, techType );

Map<String, String> NewtechType = AG\_Post\_TrainingController.getNewTechType();

system.assertNotEquals(Null, NewtechType);

Map<String, String> typeofCommodity = AG\_Post\_TrainingController.getCommodityType();

system.assertNotEquals(Null, typeofCommodity);

}

@istest

static void saveAgPostTrainingCSVTest(){

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Ag Post Training Follow up Form' limit 1];

system.debug('RT@@'+RT);

AG\_Post\_TrainingController agpost = new AG\_Post\_TrainingController();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

far2.USANDO\_NUEVA\_PRACTICA\_\_c='No';

far2.N\_mero\_correspondiente\_en\_la\_lista\_de\_t\_\_c='Genética de cultivos';

far2.Tipo\_de\_hect\_rea\_\_c='Tierra de cultivo';

far2.Pr\_ctica\_de\_gesti\_n\_Tipo\_de\_t\_cnica\_\_c='Cultivo genética';

far2.NOMBRE\_DE\_PRODUCTO\_\_c='Aguacate';

far2.Name\_of\_new\_techniques\_applied\_\_c='testing';

far2.No\_of\_hectares\_\_c=decimal.valueof(56);

far2.Date\_of\_follow\_up\_visit\_to\_farmer\_\_c='22/09/2021';

far2.Contact\_Name\_\_c=con.id;

far2.RecordTypeId = RT.Id;

finalFarmlist1.add(far2);

insert finalFarmlist1;

//AG\_Post\_TrainingController.saveAgPostTraining(finalFarmlist1);

AG\_Post\_TrainingController.insertData(JSON.serialize(finalFarmlist1));

Map<String, String> Techniques = AG\_Post\_TrainingController.getTechniques();

system.assertNotEquals(Null, Techniques);

Map<String, String> typeofHectare= AG\_Post\_TrainingController.getTypeofhectare();

system.assertNotEquals(Null, typeofHectare);

Map<String, String> techType = AG\_Post\_TrainingController.getTechType();

system.assertNotEquals(Null, techType );

Map<String, String> NewtechType = AG\_Post\_TrainingController.getNewTechType();

system.assertNotEquals(Null, NewtechType);

Map<String, String> typeofCommodity = AG\_Post\_TrainingController.getCommodityType();

system.assertNotEquals(Null, typeofCommodity);

}

}

**Post Harvest cmp:**

<aura:component controller="PostHarvest\_ProductionController" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Attribute declaration-->

<aura:attribute name="FarmerList" type="Farmers\_\_c[]"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!--Attribute declaration for picklist -->

<aura:attribute name="GenderMap" type="Map"/>

<aura:attribute name="options\_GenderMap" type="List" default="[]"/>

<aura:attribute name="CommodityTypeMap" type="Map"/>

<aura:attribute name="options\_CommodityTypeMap" type="List" default="[]"/>

<aura:attribute name="SmallHolderMap" type="Map"/>

<aura:attribute name="options\_SmallHolderMap" type="List" default="[]"/>

<aura:attribute name="TypeofLiveMap" type="Map"/>

<aura:attribute name="options\_TypeofLiveMap" type="List" default="[]"/>

<aura:attribute name="commodityMap" type="Map"/>

<aura:attribute name="options\_CommodityMap" type="List" default="[]"/>

<aura:attribute name="AgetypeMap" type="Map"/>

<aura:attribute name="options\_AgeTypeMap" type="List" default="[]"/>

<!--Attribute declaration for lookup field-->

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<aura:handler name="change" value="{!v.selectedContactId}" action="{!c.handleContactChange}" />

<aura:attribute name="selectedContactId" type="String" />

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<!--Action Method/doInit-->

<!-- <aura:handler name="init" value="{!this}" action="{!c.doInit}"/>-->

<aura:handler name="init" value="{!this}" action="{!c.doCommodityType}"/>

<aura:handler name="init" value="{!this}" action="{!c.dosmallholder}"/>

<aura:handler name="init" value="{!this}" action="{!c.doTypeofStock}"/>

<aura:handler name="init" value="{!this}" action="{!c.doNameofCommodity}"/>

<!-- <aura:handler name="init" value="{!this}" action="{!c.doAgeType}"/>-->

<aura:html tag="style">

.slds-col--padded.contentRegion.comm-layout-column{

width:1300px !important;

margin-left:-80px !important;

}

</aura:html>

<!--lightning component for designing UI-->

<div class="slds-m-around--x--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!-- <img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Times New Roman; color:green;">Formulario de producción posterior a la cosecha </span>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-cell-wrap" >S.No</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="ID" align="center" >Identificación<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-cell-wrap" title="Age " align="center">Edad<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Gender " align="center">Genero<abbr class="slds-required" >\*</abbr>

</div>

</th>-->

<th scope="col">

<div class="slds-cell-wrap" title="Commodity Type" align="center">Tipo de producto<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Commodity name " align="center">Nombre de producto <abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Total Production" align="center">Producción total<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Total Units of Production" align="center">Unidades totales de producción<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Smallholder" align="center">Pequeño productor<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Type of Livestock" align="center">Tipo de ganado

<lightning:helptext content="solo aplicable para productores ganaderos" class="customIcon"/>

</div>

</th>

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.FarmerList}" var="Far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!--<lightning:input name="ID" type="text" value="{!Far.ID\_\_c}" />-->

<!-- <c:LookupPersonId objectAPIName="contact" IconName="standard:contact" />-->

<!-- Calling another component for creating lookup -->

<c:ContactFarmerLookUp

objectAPIName="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId="{!Far.Contact\_Name\_\_c}"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<!-- <td>

<lightning:combobox name="Age" type="number" value="{!Far.Edad\_\_c}" options="{!v.options\_AgeTypeMap}" placeholder=""/>

</td>

<td>

<lightning:combobox name="Gender " type="text" value="{!Far.Genero\_\_c}" options="{!v.options\_GenderMap}" placeholder="" />

</td>-->

<td>

<lightning:combobox name="Commodity Type" type="Commodity Type" value="{!Far.TIPO\_DE\_PRODUCTO\_\_c}" options="{!v.options\_CommodityTypeMap}" placeholder="" />

</td>

<td>

<lightning:combobox name="Commodity Name" type="text" value="{!Far.NOMBRE\_DE\_PRODUCTO\_\_c}" options="{!v.options\_CommodityMap}" placeholder=""/>

</td>

<td>

<lightning:input name="Total Production " type="text" maxlength="20" value="{!Far.Total\_Production\_\_c}" />

</td>

<td>

<lightning:input name="Total Units of Production" type="text" maxlength="20" value="{!Far.Total\_Units\_of\_Production\_\_c}"/>

</td>

<td>

<lightning:combobox name="Smallholder " type="text" value="{!Far.PEQUE\_O\_PRODUCTOR\_\_c}" options="{!v.options\_SmallHolderMap}" placeholder="" />

</td>

<td>

<lightning:combobox name="Type of Livestock " type="New buyer" value="{!Far.TIPO\_DE\_GANADO\_\_c}" options="{!v.options\_TypeofLiveMap}" placeholder="" />

</td>

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 90rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario de producción posterior a la cosecha</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

</aura:component>

**Controller:**

({

// Showing the values in Popup while uploading files in both .csv and .xlsx format

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("TipoDeProducto", "NombreDeProducto", "Producciontotal", "Unidadestotalesdeproduccion", "Pequenoproductor", "Tipodeganado", "Identificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

// alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

//alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

// alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 7){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("TipoDeProducto", "NombreDeProducto", "Producciontotal", "Unidadestotalesdeproduccion", "Pequenoproductor", "Tipodeganado", "Identificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

//alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 7){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

// calling serverside/Apex comtroller for getting picklist Values

doCommodityType : function(component, event, helper) {

helper.addFarmerRecord(component, event);

var action = component.get("c.getCommodityType");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var CommodityTypeMap = [];

var TypeofCommodityMapOptions = [];

for(var key in result){

CommodityTypeMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeofCommodityMapOptions.push(state);

}

component.set("v.CommodityTypeMap", CommodityTypeMap);

component.set("v.options\_CommodityTypeMap", TypeofCommodityMapOptions);

}

});

$A.enqueueAction(action);

},

dosmallholder : function(component, event, helper) {

var action = component.get("c.getSmallHolder");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var HolderMap = [];

var HolderMapOptions = [];

for(var key in result){

HolderMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

HolderMapOptions.push(state);

}

component.set("v.SmallHolderMap",HolderMap);

component.set("v.options\_SmallHolderMap", HolderMapOptions);

}

});

$A.enqueueAction(action);

},

doTypeofStock : function(component, event, helper) {

var action = component.get("c.getTypeofStock");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var Stocktype = [];

var StocktypeMapOptions = [];

for(var key in result){

Stocktype.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

StocktypeMapOptions.push(state);

}

component.set("v.TypeofLiveMap", Stocktype);

component.set("v.options\_TypeofLiveMap", StocktypeMapOptions);

}

});

$A.enqueueAction(action);

},

doNameofCommodity : function(component, event, helper) {

var action = component.get("c.getCommodity");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var Commoditytype = [];

var CommoditytypeMapOptions = [];

for(var key in result){

Commoditytype.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

CommoditytypeMapOptions.push(state);

}

component.set("v.commodityMap", Commoditytype);

component.set("v.options\_CommodityMap", CommoditytypeMapOptions);

}

});

$A.enqueueAction(action);

},

//Select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.FarmerList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//adding new row method

addRow: function(component, event, helper) {

helper.addFarmerRecord(component, event);

},

//remove row method(single row)

removeRow: function(component, event, helper) {

//Get the Farmer list

var FarmerList = component.get("v.FarmerList");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

FarmerList.splice(index, 1);

component.set("v.FarmerList", FarmerList);

},

//final method to create records in server side

save: function(component, event, helper) {

console.log('in Controller Js-1');

if (helper.validateButton(component, event)){

console.log('in Controller Js-2');

helper.saveFarmerList(component, event);

}

else{

console.log('Cant save');

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

// Validating and calling the save method from helper

readCSV: function(component, event, helper) {

const headers2 = new Array("TipoDeProducto", "NombreDeProducto", "Producciontotal", "Unidadestotalesdeproduccion", "Pequenoproductor", "Tipodeganado", "Identificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 7){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].TipoDeProducto=="" ||

data1[k].NombreDeProducto==""||

data1[k].Producciontotal=="" ||

data1[k].Unidadestotalesdeproduccion=="" ||

data1[k].Pequenoproductor=="" ||

// data1[k].Tipodeganado=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

else if(data1[k].TipoDeProducto =="Ganado" && data1[k].Tipodeganado == "" ){

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Por favor complete tipo de Ganado " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var TipoDeProducto ;

var NombreDeProducto;

var Producciontotal;

var Unidadestotalesdeproduccion;

var Pequenoproductor;

var Tipodeganado;

var Identificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

//alert(rows);

arr.pop();

//alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 7){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].TipoDeProducto=="" ||

data1[k].NombreDeProducto==""||

data1[k].Producciontotal=="" ||

data1[k].Unidadestotalesdeproduccion=="" ||

data1[k].Pequenoproductor=="" ||

//data1[k].Tipodeganado=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

else if(data1[k].TipoDeProducto =="Ganado" && data1[k].Tipodeganado == "" ){

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Por favor complete tipo de Ganado " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var TipoDeProducto ;

var NombreDeProducto;

var Producciontotal;

var Unidadestotalesdeproduccion;

var Pequenoproductor;

var Tipodeganado;

var Identificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

CSV2JSON: function (component,csv) {

// console.log('Incoming csv = ' + csv);

//var array = [];

var arr = [];

arr = csv.split('\n');

//console.log('Array = '+array);

// console.log('arr = '+arr);

arr.pop();

var jsonObj = [];

var headers = arr[0].split(',');

for(var i = 1; i < arr.length; i++) {

var data = arr[i].split(',');

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

jsonObj.push(obj);

}

var data1 = jsonObj;

for( var k=0;k<jsonObj.length;k++){

console.log(data1);

if(data1[k].TipoDeProducto=="" ||

data1[k].NombreDeProducto==""||

data1[k].Producciontotal=="" ||

data1[k].Unidadestotalesdeproduccion=="" ||

data1[k].Pequenoproductor=="" ||

// data1[k].Tipodeganado=="" ||

data1[k].Identificacion==""){

// alert('Please check Provided values are Null');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

else if(data1[k].TipoDeProducto =="Ganado" && data1[k].Tipodeganado == "" ){

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Por favor complete los detalles del ganado " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

/\* isValid = false;

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

// return isValid;

}

}

return isValid;\*/

var json = JSON.stringify(data1);

console.log('json = '+ json);

return json;

},

CreateContact : function (component,jsonstr){

// console.log('jsonstr' + jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

//alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

// alert('result callback>>'+ result);

// alert("Records Inserted Succesfully");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

addFarmerRecord: function(component, event) {

//get the Farmer List from component

var FarmerList = component.get("v.FarmerList");

//Add New Farmer Record

FarmerList.push({

'sobjectType': 'Farmers\_\_c',

'TIPO\_DE\_PRODUCTO\_\_c':'',

'NOMBRE\_DE\_PRODUCTO\_\_c':'',

'PEQUE\_O\_PRODUCTOR\_\_c':'',

'Total\_Production\_\_c':'',

'Total\_Units\_of\_Production\_\_c':'',

'TIPO\_DE\_GANADO\_\_c':'',

'Contact\_Name\_\_c' : '',

});

component.set("v.FarmerList", FarmerList);

},

//Process the selected Farmers records to remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.FarmerList");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.FarmerList", finalFarmers);

},

//to create a record in server side

saveFarmerList: function(component, event, helper) {

//Call Apex class and pass Farmer list parameters

var action = component.get("c.savePostHarvestForm");

action.setParams({

"Farmlist": component.get("v.FarmerList"),

});

console.log('action1>>'+ action);

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

component.set("v.FarmerList", []);

console.log('state>>'+ state);

// alert('Registros guardados ');//eng translation for alert:Records Saved Successfully

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action);

},

//checking Validation for all required field

validateButton: function(component, event, helper) {

//Validate all account records

var isValid = true;

var farmersList = component.get("v.FarmerList");

for (var i = 0; i < farmersList.length; i++) {

if (farmersList[i].Contact\_Name\_\_c == '' ||

// farmersList[i].Edad\_\_c == '' ||

// farmersList[i].Genero\_\_c == '' ||

farmersList[i].TIPO\_DE\_PRODUCTO\_\_c == '' ||

farmersList[i].NOMBRE\_DE\_PRODUCTO\_\_c == '' ||

farmersList[i].PEQUE\_O\_PRODUCTOR\_\_c == '' ||

farmersList[i].Total\_Production\_\_c == '' ||

farmersList[i].Total\_Units\_of\_Production\_\_c == '' )

{

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila "+(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

else if (farmersList[i].TIPO\_DE\_PRODUCTO\_\_c == 'Ganado' && $A.util.isEmpty(farmersList[i].TIPO\_DE\_GANADO\_\_c) ) {

isValid = false;

// alert('Por favor complete los detalles del ganado');// please fill the livestock details

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Advertencia!",

"message": "Por favor complete los detalles del ganado",

duration:' 3000',

key: 'info\_alt',

type: 'warning',

mode: 'pester'

});

toastEvent.fire();

}

}

return isValid;

},

})

**Css:**

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public class PostHarvest\_ProductionController {

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list<fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Post Harvest Production Form' limit 1];

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.TIPO\_DE\_PRODUCTO\_\_c = wrapper.TipoDeProducto;

far1.NOMBRE\_DE\_PRODUCTO\_\_c = wrapper.NombreDeProducto;

far1.Total\_Production\_\_c = wrapper.Producciontotal;

far1.Total\_Units\_of\_Production\_\_c = wrapper.Unidadestotalesdeproduccion;

far1.PEQUE\_O\_PRODUCTOR\_\_c = wrapper.Pequenoproductor;

far1.TIPO\_DE\_GANADO\_\_c = wrapper.Tipodeganado;

if(wrapper.Identificacion!= null){

far1.Code\_\_c = wrapper.Identificacion;

}

ListtoInsert.add(far1);

}

if(ListtoInsert.size() > 0){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String TipoDeProducto;

public String NombreDeProducto;

public String Producciontotal;

public String Unidadestotalesdeproduccion;

public String Pequenoproductor;

public String Tipodeganado;

public String Identificacion;

}

@AuraEnabled

public static void savePostHarvestForm(list<Farmers\_\_c> Farmlist){

// Insert Farmlist;

system.debug('Farmlist>>'+Farmlist);

list<Farmers\_\_c> finalFarmlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Post Harvest Production Form' limit 1];

for(Farmers\_\_c far : Farmlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmlist.add(far);

}

system.debug('finalFarmlist>>'+finalFarmlist);

Insert finalFarmlist;

/\* try{

Insert Farmlist;

return 'SUCCESS';

} catch(Exception e) {

System.debug('The following exception has occurred: ' +e.getLineNumber()+ e.getMessage());

return e.getMessage();

}\*/

}

@AuraEnabled

public static Map<String, String> getCommodityType(){

Map<String, String> optionsCommodityTypeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.TIPO\_DE\_PRODUCTO\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsCommodityTypeMap.put(ple.getValue(), ple.getLabel());

}

return optionsCommodityTypeMap;

}

@AuraEnabled

public static Map<String, String> getSmallHolder(){

Map<String, String> optionsSmallHolderMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.PEQUE\_O\_PRODUCTOR\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsSmallHolderMap.put(ple.getValue(), ple.getLabel());

}

return optionsSmallHolderMap;

}

@AuraEnabled

public static Map<String, String> getTypeofStock(){

Map<String, String> optionsStockMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.TIPO\_DE\_GANADO\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsStockMap.put(ple.getValue(), ple.getLabel());

}

return optionsStockMap;

}

@AuraEnabled

public static Map<String, String> getCommodity(){

Map<String, String> optionsCommodityMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.NOMBRE\_DE\_PRODUCTO\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsCommodityMap.put(ple.getValue(), ple.getLabel());

}

return optionsCommodityMap;

}

/\* @AuraEnabled

public static Map<String, String> getAgeMap(){

Map<String, String> optionsAgeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Edad\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsAgeMap.put(ple.getValue(), ple.getLabel());

}

return optionsAgeMap;

}

@AuraEnabled

public static Map<String, String> getGender(){

Map<String, String> optionsGenderMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Genero\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsGenderMap.put(ple.getValue(), ple.getLabel());

}

return optionsGenderMap;

}\*/

}

**TestClass:**

@istest

public class PostHarvest\_ProductionControllerTest {

@istest

static void savePostHarvestFormTest(){

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Post Harvest Production Form' limit 1];

system.debug('RT@@'+RT);

PostHarvest\_ProductionController posthar = new PostHarvest\_ProductionController();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

far1.TIPO\_DE\_PRODUCTO\_\_c='Cultivo';

far1.NOMBRE\_DE\_PRODUCTO\_\_c='Ajo';

far1.PEQUE\_O\_PRODUCTOR\_\_c='sí';

far1.Total\_Production\_\_c='4';

far1.Total\_Units\_of\_Production\_\_c='5';

far1.TIPO\_DE\_GANADO\_\_c='Pastizales';

far1.Contact\_Name\_\_c=con.id;

far1.RecordTypeId = RT.Id;

insert finalFarmlist1;

Test.startTest();

PostHarvest\_ProductionController.savePostHarvestForm(finalFarmlist1);

Map<String, String> Commoditytype = PostHarvest\_ProductionController.getCommodityType();

system.assertNotEquals(Null, Commoditytype);

Map<String, String> Smallholder= PostHarvest\_ProductionController.getSmallHolder();

system.assertNotEquals(Null, Smallholder);

Map<String, String> TypeofStock = PostHarvest\_ProductionController.getTypeofStock();

system.assertNotEquals(Null, TypeofStock );

Map<String, String> Commodity = PostHarvest\_ProductionController.getCommodity();

system.assertNotEquals(Null, Commodity);

Test.stopTest();

}

@istest

static void savePostHarvestFormCSVTest(){

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Post Harvest Production Form' limit 1];

system.debug('RT@@'+RT);

PostHarvest\_ProductionController posthar = new PostHarvest\_ProductionController();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

far2.TIPO\_DE\_PRODUCTO\_\_c='Cultivo';

far2.NOMBRE\_DE\_PRODUCTO\_\_c='Ajo';

far2.PEQUE\_O\_PRODUCTOR\_\_c='sí';

far2.Total\_Production\_\_c='4';

far2.Total\_Units\_of\_Production\_\_c='5';

far2.TIPO\_DE\_GANADO\_\_c='Pastizales';

far2.Contact\_Name\_\_c=con.id;

far2.RecordTypeId = RT.Id;

finalFarmlist1.add(far2);

insert finalFarmlist1;

//PostHarvest\_ProductionController.savePostHarvestForm(finalFarmlist1);

PostHarvest\_ProductionController.insertData(JSON.serialize(finalFarmlist1));

Map<String, String> Commoditytype = PostHarvest\_ProductionController.getCommodityType();

system.assertNotEquals(Null, Commoditytype);

Map<String, String> Smallholder= PostHarvest\_ProductionController.getSmallHolder();

system.assertNotEquals(Null, Smallholder);

Map<String, String> TypeofStock = PostHarvest\_ProductionController.getTypeofStock();

system.assertNotEquals(Null, TypeofStock );

Map<String, String> Commodity = PostHarvest\_ProductionController.getCommodity();

system.assertNotEquals(Null, Commodity);

}

}

**Annual sales cmp:**

<aura:component controller="AnnualSalesForm\_Controller" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global">

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Attribute declaration-->

<aura:attribute name="farmersList" type="Farmers\_\_c[]"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!--Attribute declaration for lookup feild-->

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<aura:handler name="change" value="{!v.selectedContactId}" action="{!c.handleContactChange}" />

<aura:attribute name="selectedContactId" type="String" />

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<!--Attributes for picKlist values-->

<aura:attribute name="GenderFOMap" type="Map"/>

<aura:attribute name="options\_GenderFO" type="List" default="[]"/>

<aura:attribute name="AgeFOMap" type="Map"/>

<aura:attribute name="options\_AgeFO" type="List" default="[]"/>

<aura:attribute name="TypeOfProdMap" type="Map"/>

<aura:attribute name="options\_TypeOfProd" type="List" default="[]"/>

<aura:attribute name="TypeOfProducerMap" type="Map"/>

<aura:attribute name="options\_TypeOfProducer" type="List" default="[]"/>

<aura:attribute name="CurrencyMap" type="Map"/>

<aura:attribute name="options\_Currency" type="List" default="[]"/>

<!--Action Method/doInIt-->

<aura:handler name="init" value="{!this}" action="{!c.addRow}"/>

<aura:html tag="style">

.slds-col--padded.contentRegion.comm-layout-column{

width:1300px !important;

margin-left:-80px !important;

}

</aura:html>

<!--lightning components for designing UI-->

<div class="slds-m-around--x--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<!--div class="slds-p-bottom\_large" align="center"-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Arial; color:green;">Venta anual</span>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button iconName="utility:add"

label="Agregar Fila"

title="Click to add rows"

variant="brand"

onclick="{!c.addRow}"/>

<lightning:button iconName="utility:delete"

label="Quitar Fila"

title="Remove Selected Rows"

variant="destructive"

onclick="{!c.handleConfirmDialog}"/>

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<!--table class="slds-table slds-text-align\_center slds-table\_cell-buffer slds-table\_bordered slds-table\_fixed-layout slds-table\_striped"-->

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-cell-wrap ">S.No</div>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="ID" >Identificación<abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Name of firm or association" >Nombre de firma o asociación<abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center>

<div class="slds-cell-wrap" title="Gender of firm owner" >Genero del dueño de la firma<abbr class="slds-required" >\*</abbr>

<lightning:helptext content="Seleccione 'mezclado' si la firma tiene más de un dueño" class="customIcon"/>

</div>

</center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Age of Firm owner" >Edad del dueño de La firma<abbr class="slds-required" >\*</abbr>

<lightning:helptext content="Seleccione 'mezclado' si la firma tiene más de un dueño" class="customIcon"/>

</div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Name of Product" >Nombre de producto<abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Select Type of product" >Tipo de producto <abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Type of Producer" >Tipo de productor<abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Cash value of commodities sold" >Valor de Productos vendidas <abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Currency" >Moneda <abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Date sold" >Fecha de venta<abbr class="slds-required" >\*</abbr></div></center>

</th>

<!--th scope="col">

<div class="slds-cell-wrap" title="Action">Action</div>

</th-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.farmersList}" var="far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!--lightning:input name="ID" type="text" value="{!far.ID\_\_c}" /-->

<!-- <c:LookupPersonId objectAPIName="contact" IconName="standard:contact" />-->

<!-- Calling another component for creating lookup -->

<c:ContactFarmerLookUp

objectAPIName="Contact"

label="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId ="{!far.Contact\_Name\_\_c}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<td>

<lightning:input name="farName" type="text" maxlength="50" value="{!far.Name\_of\_firm\_or\_association\_\_c}" variant="label-hidden" />

</td>

<td>

<lightning:combobox name="farGenOwner"

type="text"

value="{!far.GENERO\_DE\_DUE\_O\_DE\_LA\_FIRMA\_\_c}"

options="{!v.options\_GenderFO}"

variant="label-hidden"

class="slds"

placeholder=""/>

</td>

<td>

<lightning:combobox name="farAgeOwner"

type="text"

value="{!far.EDAD\_DE\_DUE\_O\_DE\_FIRMA\_\_c}"

options="{!v.options\_AgeFO}"

variant="label-hidden"

class="slds"

placeholder=""/>

</td>

<td>

<lightning:input name="farNoP" type="text" value="{!far.Name\_of\_Product\_\_c}" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="farToP"

type="text"

value="{!far.TIPO\_DE\_PRODUCT\_\_c}"

options="{!v.options\_TypeOfProd}"

variant="label-hidden"

class="slds"

placeholder=""/>

</td>

<td>

<lightning:combobox name="farToPr"

type="text"

value="{!far.TIPO\_DE\_PRODUCTOR\_\_c}"

options="{!v.options\_TypeOfProducer}"

variant="label-hidden"

class="slds"

placeholder=""/>

</td>

<td>

<lightning:input name="farCashValCom" type="number" maxlength="10" value="{!far.Cash\_value\_of\_commodities\_sold\_\_c}" variant="label-hidden" class="message" />

</td>

<td>

<lightning:combobox name="farCurrency"

type="text"

value="{!far.Currency\_\_c}"

options="{!v.options\_Currency}"

variant="label-hidden"

class="slds"

placeholder=""/>

</td>

<td>

<!--<lightning:input name="farDtSold" type="date" value="{!far.Date\_sold\_\_c}" variant="label-hidden" dateStyle="short"/>-->

<lightning:input type="text" name="input" pattern="^((0[1-9])|([12][0-9])|(3[01]))\/((0[1-9])|(1[0-2]))\/((19[0-9]{2})|(2[0-9]{3}))$" required="true" value="{!far.Date\_sold\_\_c}" class="date" placeholder="DD/MM/AAAA" variant="label-hidden"/>

</td>

<!--td>

<a onclick="{!c.removeRecord}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td-->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.saveFarmers}" />

</div>

</div>

</div>

</div>

</div>

</div>

<!--/div-->

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 70rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Venta anual</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

<!--/div-->

</aura:component>

**Controller:**

({

addRow: function(component, event, helper) {

//getting picklist values from helper

helper.getGenderPicVals(component, event, helper);

helper.getAgePicVals(component, event, helper);

helper.getTypeOfProdPicVals(component, event, helper);

helper.getProducerPicVals(component, event, helper);

helper.getCurrencyPicVals(component, event, helper);

//get the account List from component

var farmersList = component.get("v.farmersList");

//Add New Account Record

farmersList.push({

"sobjectType": "Farmers\_\_c",

'Name\_of\_firm\_or\_association\_\_c': '',

'GENERO\_DE\_DUE\_O\_DE\_LA\_FIRMA\_\_c': '',

'EDAD\_DE\_DUE\_O\_DE\_FIRMA\_\_c': '',

'Name\_of\_Product\_\_c': '',

'TIPO\_DE\_PRODUCT\_\_c': '',

'TIPO\_DE\_PRODUCTOR\_\_c': '',

'Cash\_value\_of\_commodities\_sold\_\_c': '',

'Currency\_\_c': '',

'Date\_sold\_\_c': null,

'Contact\_Name\_\_c' : ''

});

// alert('farmersList'+farmersList.GENERO\_DE\_DUE\_O\_DE\_LA\_FIRMA\_\_c);

component.set("v.farmersList", farmersList);

},

//Select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.farmersList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//remove row method (single row)

removeRecord: function(component, event, helper) {

//Get the farmer list

var farmersList = component.get("v.farmersList");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

//Remove single record from farmer list

farmersList.splice(index, 1);

//Set modified farmer list

component.set("v.farmersList", farmersList);

},

//final method to create records in server side

saveFarmers: function(component, event, helper) {

if (helper.validateFarmerRecords(component, event)) {

//Call Apex method and pass farmer list as a parameters

var action = component.get("c.saveFarmerRecords");

action.setParams({

"farmerList": component.get("v.farmersList")

});

action.setCallback(this, function(response) {

//get response status

var state = response.getState();

if (state === "SUCCESS") {

//set empty farmer list

component.set("v.farmersList", []);

// alert('Registros guardados');//eng translation for alert:Records Saved Successfully

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '20000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action);

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

//New code implementation for uploading csv and excel

// Showing the values in Popup while uploading files in both .csv and .xlsx format

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("Nombredefirmaoasociacion", "Generofirma", "Edadfirma", "Nombredeproducto", "Tipodeproducto", "Tipodeproductor", "ValordeProductosvendidas", "Moneda", "Fechadeventa", "Identificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

// alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

//alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

// alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 10){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("Nombredefirmaoasociacion", "Generofirma", "Edadfirma", "Nombredeproducto", "Tipodeproducto", "Tipodeproductor", "ValordeProductosvendidas", "Moneda", "Fechadeventa", "Identificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

//alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 10){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

// Validating and Saving the records while uploading both .csv and .xlsx format

readCSV: function(component, event, helper) {

const headers2 = new Array("Nombredefirmaoasociacion", "Generofirma", "Edadfirma", "Nombredeproducto", "Tipodeproducto", "Tipodeproductor", "ValordeProductosvendidas", "Moneda", "Fechadeventa", "Identificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 10){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].Nombredefirmaoasociacion=="" ||

data1[k].Generofirma==""||

data1[k].Edadfirma=="" ||

data1[k].Nombredeproducto=="" ||

data1[k].Tipodeproducto=="" ||

data1[k].Tipodeproductor=="" ||

data1[k]. ValordeProductosvendidas=="" ||

data1[k].Moneda=="" ||

data1[k].Fechadeventa=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

//helper.CreateContact(component, readerxlx);

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var Nombredefirmaoasociacion ;

var Generofirma;

var Edadfirma;

var Nombredeproducto;

var Tipodeproducto;

var Tipodeproductor;

var ValordeProductosvendidas;

var Moneda;

var Fechadeventa;

var Identificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

//alert(rows);

arr.pop();

//alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 10){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].Nombredefirmaoasociacion=="" ||

data1[k].Generofirma==""||

data1[k].Edadfirma=="" ||

data1[k].Nombredeproducto=="" ||

data1[k].Tipodeproducto=="" ||

data1[k].Tipodeproductor=="" ||

data1[k]. ValordeProductosvendidas=="" ||

data1[k].Moneda=="" ||

data1[k].Fechadeventa=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var Nombredefirmaoasociacion ;

var Generofirma;

var Edadfirma;

var Nombredeproducto;

var Tipodeproducto;

var Tipodeproductor;

var ValordeProductosvendidas;

var Moneda;

var Fechadeventa;

var Identificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

})

**Helper:**

({

//checking validation for all required fields

validateFarmerRecords: function(component, event) {

//Validate all account records

var isValid = true;

var farmersList = component.get("v.farmersList");

for (var i = 0; i < farmersList.length; i++) {

if (farmersList[i].Name\_of\_firm\_or\_association\_\_c == '' ||

farmersList[i].Contact\_Name\_\_c == '' ||

farmersList[i].GENERO\_DE\_DUE\_O\_DE\_LA\_FIRMA\_\_c == '' ||

farmersList[i].EDAD\_DE\_DUE\_O\_DE\_FIRMA\_\_c == '' ||

farmersList[i].Name\_of\_Product\_\_c == '' ||

farmersList[i].TIPO\_DE\_PRODUCT\_\_c == '' ||

farmersList[i].TIPO\_DE\_PRODUCTOR\_\_c == '' ||

farmersList[i].Cash\_value\_of\_commodities\_sold\_\_c == '' ||

farmersList[i].Currency\_\_c == '' ||

farmersList[i].Date\_sold\_\_c == '') {

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1)); eng translation is mandatory feilds cannot be blank

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila "+(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

//Process the selected Farmers records for remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.farmersList");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.farmersList", finalFarmers);

},

//calling server side/apex controller to getting picklist Values

getGenderPicVals : function(component) {

var action = component.get("c.getGender");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var GenderFOMap = [];

var optionsGenderFO = [];

for(var key in result){

GenderFOMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

optionsGenderFO.push(state);

}

component.set("v.GenderFOMap", GenderFOMap);

component.set("v.options\_GenderFO", optionsGenderFO);

}

});

$A.enqueueAction(action);

},

getAgePicVals : function(component) {

var action = component.get("c.getAgeMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var AgeFOMap = [];

var optionsAgeFO = [];

for(var key in result){

AgeFOMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

optionsAgeFO.push(state);

}

component.set("v.AgeFOMap", AgeFOMap);

component.set("v.options\_AgeFO", optionsAgeFO);

}

});

$A.enqueueAction(action);

},

getTypeOfProdPicVals : function(component) {

var action = component.get("c.getTipoProdt");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeOfProdMap = [];

var optionsTypeOfProd = [];

for(var key in result){

TypeOfProdMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

optionsTypeOfProd.push(state);

}

component.set("v.TypeOfProdMap", TypeOfProdMap);

component.set("v.options\_TypeOfProd", optionsTypeOfProd);

}

});

$A.enqueueAction(action);

},

getProducerPicVals : function(component) {

var action = component.get("c.TipoProdr");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeOfProducerMap = [];

var optionsTypeOfProducer = [];

for(var key in result){

TypeOfProducerMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

optionsTypeOfProducer.push(state);

}

component.set("v.TypeMap", TypeOfProducerMap);

component.set("v.options\_TypeOfProducer", optionsTypeOfProducer);

}

});

$A.enqueueAction(action);

},

getCurrencyPicVals : function(component) {

var action = component.get("c.getCurrency");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var CurrencyMap = [];

var optionsCurrency = [];

for(var key in result){

CurrencyMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

optionsCurrency.push(state);

}

component.set("v.CurrencyMap", CurrencyMap);

component.set("v.options\_Currency", optionsCurrency);

}

});

$A.enqueueAction(action);

},

// New code for uploading csv and excel

//Saving the records while uploading files

CreateContact : function (component,jsonstr){

// alert('inside serverside');

// console.log('jsonstr' + jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

// alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

//alert('result callback>>'+ result);

//alert("Records Inserted Succesfully");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

})

**Css:**

/\*.THIS .slds-truncate {

text-align: center;

}\*/

.THIS .slds-icon-utility-delete.slds-icon\_container{

padding-top: 14px;

padding-left: 18px;

}

.THIS .date .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .date .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"Utilice el siguiente formato: DD/MM/AAAA";

}

.THIS .date .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .message .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .message .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"Ingrese un valor válido";

}

.THIS .message .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .slds{

width: 100px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public with sharing class AnnualSalesForm\_Controller {

@AuraEnabled

public static void saveFarmerRecords(List<Farmers\_\_c> farmerList){

//Insert farmerList;

system.debug('Farmlist>>'+farmerlist);

list<Farmers\_\_c> finalFarmlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Annual Sales' limit 1];

for(Farmers\_\_c far : farmerlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmlist.add(far);

}

system.debug('finalFarmlist>>'+finalFarmlist);

Insert finalFarmlist;

}

@AuraEnabled

public static Map<String, String> getGender(){

Map<String, String> optionsGender = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.GENERO\_DE\_DUE\_O\_DE\_LA\_FIRMA\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsGender.put(ple.getValue(), ple.getLabel());

}

return optionsGender;

}

@AuraEnabled

public static Map<String, String> getAgeMethod(){

Map<String, String> optionsFirma = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.EDAD\_DE\_DUE\_O\_DE\_FIRMA\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsFirma.put(ple.getValue(), ple.getLabel());

}

return optionsFirma;

}

@AuraEnabled

public static Map<String, String> getTipoProdt(){

Map<String, String> optionsTipoProdt = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.TIPO\_DE\_PRODUCT\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsTipoProdt.put(ple.getValue(), ple.getLabel());

}

return optionsTipoProdt;

}

@AuraEnabled

public static Map<String, String> TipoProdr(){

Map<String, String> optionsTipoProdr = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.TIPO\_DE\_PRODUCTOR\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsTipoProdr.put(ple.getValue(), ple.getLabel());

}

return optionsTipoProdr;

}

@AuraEnabled

public static Map<String, String> getCurrency(){

Map<String, String> optionsCurrency = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Currency\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsCurrency.put(ple.getValue(), ple.getLabel());

}

return optionsCurrency;

}

// new code for upload csv

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list< fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Annual Sales' limit 1];

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.Name\_of\_firm\_or\_association\_\_c = wrapper.Nombredefirmaoasociacion;

far1.GENERO\_DE\_DUE\_O\_DE\_LA\_FIRMA\_\_c = wrapper.Generofirma;

far1.EDAD\_DE\_DUE\_O\_DE\_FIRMA\_\_c = wrapper.Edadfirma;

far1.Name\_of\_Product\_\_c = wrapper.Nombredeproducto;

far1.TIPO\_DE\_PRODUCT\_\_c = wrapper.Tipodeproducto;

far1.TIPO\_DE\_PRODUCTOR\_\_c = wrapper.Tipodeproductor;

far1.Cash\_value\_of\_commodities\_sold\_\_c = wrapper.ValordeProductosvendidas;

far1.Currency\_\_c = wrapper.Moneda;

far1.Date\_sold\_\_c = wrapper.Fechadeventa;

if(wrapper.Identificacion!= null){

far1.Code\_\_c = wrapper.Identificacion;

}

ListtoInsert.add(far1);

}

if(ListtoInsert.size() > 0){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String Nombredefirmaoasociacion;

public String Generofirma;

public String Edadfirma;

public String Nombredeproducto;

public String Tipodeproducto;

public String Tipodeproductor;

public Decimal ValordeProductosvendidas;

public String Moneda;

public String Fechadeventa;

public String Identificacion;

}

}

**TestClass:**

@isTest

public class AnnualSalesForm\_ControllerTest {

@istest

static void AnnualSalesTest(){

string value;

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Annual Sales' limit 1];

system.debug('RT@@'+RT);

AnnualSalesForm\_Controller ag = new AnnualSalesForm\_Controller();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

far1.Name\_of\_firm\_or\_association\_\_c='test';

far1.GENERO\_DE\_DUE\_O\_DE\_LA\_FIRMA\_\_c ='Mujer';

far1.EDAD\_DE\_DUE\_O\_DE\_FIRMA\_\_c='30 y más';

far1.Name\_of\_Product\_\_c='test';

far1.TIPO\_DE\_PRODUCT\_\_c='Productos agrícolas';

far1.TIPO\_DE\_PRODUCTOR\_\_c ='minifundista (menos de 5 hectáreas en las que están trabajando)';

far1.Cash\_value\_of\_commodities\_sold\_\_c=1234;

far1.Currency\_\_c= 'USD';

far1.Date\_sold\_\_c = '12/02/2022';

far1.Contact\_Name\_\_c= con.id;

insert far1;

// far1.RecordTypeId = RT.Id;

// value = far1.GENERO\_DE\_DUE\_O\_DE\_LA\_FIRMA\_\_c;

//finalFarmlist1.add(far1);

insert finalFarmlist1;

AnnualSalesForm\_Controller.saveFarmerRecords(finalFarmlist1);

Map<String, String> getGender = AnnualSalesForm\_Controller.getGender();

system.assertNotEquals(Null, getGender);

Map<String, String> getAgeMethod = AnnualSalesForm\_Controller.getAgeMethod();

system.assertNotEquals(Null, getAgeMethod);

Map<String, String> getTipoProdt = AnnualSalesForm\_Controller.getTipoProdt();

system.assertNotEquals(Null, getTipoProdt);

Map<String, String> TipoProdr = AnnualSalesForm\_Controller.TipoProdr();

system.assertNotEquals(Null, TipoProdr);

Map<String, String> getCurrency = AnnualSalesForm\_Controller.getCurrency();

system.assertNotEquals(Null, getCurrency);

}

@istest

static void AnnualSalesCSVTest(){

string value;

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Annual Sales' limit 1];

system.debug('RT@@'+RT);

AnnualSalesForm\_Controller ag = new AnnualSalesForm\_Controller();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

far2.Name\_of\_firm\_or\_association\_\_c='test';

far2.GENERO\_DE\_DUE\_O\_DE\_LA\_FIRMA\_\_c ='Mujer';

far2.EDAD\_DE\_DUE\_O\_DE\_FIRMA\_\_c='30 y más';

far2.Name\_of\_Product\_\_c='test';

far2.TIPO\_DE\_PRODUCT\_\_c='Productos agrícolas';

far2.TIPO\_DE\_PRODUCTOR\_\_c ='minifundista (menos de 5 hectáreas en las que están trabajando)';

far2.Cash\_value\_of\_commodities\_sold\_\_c=1234;

far2.Currency\_\_c= 'USD';

far2.Date\_sold\_\_c = '12/02/2022';

far2.Contact\_Name\_\_c= con.id;

//insert far2;

// far1.RecordTypeId = RT.Id;

// value = far1.GENERO\_DE\_DUE\_O\_DE\_LA\_FIRMA\_\_c;

finalFarmlist1.add(far2);

insert finalFarmlist1;

AnnualSalesForm\_Controller.insertData(JSON.serialize(finalFarmlist1));

// AnnualSalesForm\_Controller.saveFarmerRecords(finalFarmlist1);

Map<String, String> getGender = AnnualSalesForm\_Controller.getGender();

system.assertNotEquals(Null, getGender);

Map<String, String> getAgeMethod = AnnualSalesForm\_Controller.getAgeMethod();

system.assertNotEquals(Null, getAgeMethod);

Map<String, String> getTipoProdt = AnnualSalesForm\_Controller.getTipoProdt();

system.assertNotEquals(Null, getTipoProdt);

Map<String, String> TipoProdr = AnnualSalesForm\_Controller.TipoProdr();

system.assertNotEquals(Null, TipoProdr);

Map<String, String> getCurrency = AnnualSalesForm\_Controller.getCurrency();

system.assertNotEquals(Null, getCurrency);

}

}

**Commodity Sales cmp:**

<aura:component controller="Commodity\_Sales\_Form\_Controller" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Attribute Declaration-->

<aura:attribute name="FarmerList" type="Farmers\_\_c[]"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!--Attribute declaration for picklist values-->

<aura:attribute name="CommodityMap" type="Map"/>

<aura:attribute name="options\_CommodityMap" type="List" default="[]"/>

<aura:attribute name="CommodityTypeMap" type="Map"/>

<aura:attribute name="options\_CommodityTypeMap" type="List" default="[]"/>

<aura:attribute name="NewbuyerMap" type="Map"/>

<aura:attribute name="options\_NewbuyerMap" type="List" default="[]"/>

<aura:attribute name="typeAgeMap" type="Map"/>

<aura:attribute name="options\_typeAgeMap" type="List" default="[]"/>

<aura:attribute name="GendertypeMap" type="Map"/>

<aura:attribute name="options\_GendertypeMap" type="List" default="[]"/>

<!--Attribute declaration for lookup field-->

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<aura:handler name="change" value="{!v.selectedContactId}" action="{!c.handleContactChange}" />

<aura:attribute name="selectedContactId" type="String" />

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<!--Action Method/doInit-->

<aura:handler name="init" value="{!this}" action="{!c.doInit}"/>

<aura:handler name="init" value="{!this}" action="{!c.doCommodityType}"/>

<aura:handler name="init" value="{!this}" action="{!c.doNewBuyer}"/>

<!-- <aura:handler name="init" value="{!this}" action="{!c.doAgeTypes}"/>

<aura:handler name="init" value="{!this}" action="{!c.doGenderTypes}"/>-->

<!--lightning components for designing UI-->

<div class="slds-m-around--x--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!-- <img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Times New Roman; color:green;">Formulario de venta de producto</span>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-cell-wrap" >S.No</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="ID" align="center" >Nombre/Identificación<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-cell-wrap" title="Age" align="center" >Edad<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Gender" align="center" >Genero<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<div class="slds-cell-wrap" title="Name of Commodity " align="center">Nombre de producto <abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Commodity Type " align="center">Tipo de producto <abbr class="slds-required" >\*</abbr>

</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Name of buyer" align="center">Nombre de comprador<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Is this a new buyer? " align="center">Es un nuevo comprador? <abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="Action" align="center">Action</div>

</th>-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.FarmerList}" var="Far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!-- <lightning:input name="ID" type="text" value="{!Far.ID\_\_c}" />-->

<!-- <c:LookupPersonId objectAPIName="contact" IconName="standard:contact" />-->

<!--calling another component for creating lookup-->

<c:ContactFarmerLookUp

objectAPIName="Contact"

label="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId ="{!Far.Contact\_Name\_\_c}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<!-- <td>

<lightning:combobox name="Age" type="number" value="{!Far.Edad\_\_c}" options="{!v.options\_typeAgeMap}" placeholder="" variant="label-hidden" />

</td>

<td>

<lightning:combobox name="Gender" type="text" value="{!Far.Genero\_\_c}" options="{!v.options\_GendertypeMap}" placeholder="" variant="label-hidden"/>

</td>-->

<td>

<lightning:combobox name="Name Of Commodity" type="Commodity" value="{!Far.NOMBRE\_DE\_PRODUCTO\_\_c}" options="{!v.options\_CommodityMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="Commodity Type " type="Technique" value="{!Far.TIPO\_DE\_PRODUCTO\_\_c}" options="{!v.options\_CommodityTypeMap}" placeholder="" variant="label-hidden" />

</td>

<td>

<lightning:input name="Name of buyer" type="text" value="{!Far.Name\_of\_Buyer\_\_c}" variant="label-hidden" />

</td>

<td>

<lightning:combobox name="Is this a new buyer? Y/N " type="New buyer" value="{!Far.Es\_esta\_nueva\_compradora\_\_c}" options="{!v.options\_NewbuyerMap}" placeholder="" variant="label-hidden" />

</td>

<!-- <td>

<a onclick="{!c.removeRow}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td> -->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

<!-- <lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>-->

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 70rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario de venta de producto</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

</aura:component>

**Controller:**

({

// Showing the values in Popup while uploading files in both .csv and .xlsx format

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("Tipodeproducto", "Nombredeproducto", "Nombredecomprador", "Esunnuevocomprado", "Identificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

// alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

//alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

// alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("Tipodeproducto", "Nombredeproducto", "Nombredecomprador", "Esunnuevocomprado", "Identificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

//alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

//calling serverside/apex controller for getting PicklistValues-->

doInit : function(component, event, helper) {

helper.addFarmerRecord(component, event);

var action = component.get("c.getCommodity");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var CommodityMap = [];

var CommodityMapOptions = [];

for(var key in result){

CommodityMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

CommodityMapOptions.push(state);

// console.log(CommodityMapOptions);

//system.debug('CommodityMapOptions'+JSON.Stringfy(CommodityMapOptions));

}

component.set("v.CommodityMap", CommodityMap);

component.set("v.options\_CommodityMap", CommodityMapOptions);

}

});

$A.enqueueAction(action);

},

doCommodityType : function(component, event, helper) {

var action = component.get("c.getCommodityType");

console.log('action>>' + action);

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var CommodityTypeMap = [];

var TypeofCommodityMapOptions = [];

for(var key in result){

CommodityTypeMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeofCommodityMapOptions.push(state);

}

component.set("v.CommodityTypeMap", CommodityTypeMap);

component.set("v.options\_CommodityTypeMap", TypeofCommodityMapOptions);

}

});

$A.enqueueAction(action);

},

doNewBuyer : function(component, event, helper) {

var action = component.get("c.getNewBuyer");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var BuyerMap = [];

var TypeBuyerMapOptions = [];

for(var key in result){

BuyerMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeBuyerMapOptions.push(state);

}

component.set("v.NewbuyerMap", BuyerMap);

component.set("v.options\_NewbuyerMap", TypeBuyerMapOptions);

}

});

$A.enqueueAction(action);

},

//Select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.FarmerList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//adding a new row method

addRow: function(component, event, helper) {

helper.addFarmerRecord(component, event);

},

//remove row method(single row)

removeRow: function(component, event, helper) {

//Get the Farmer list

var FarmerList = component.get("v.FarmerList");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

FarmerList.splice(index, 1);

component.set("v.FarmerList", FarmerList);

},

//final method to create record in server side

save: function(component, event, helper) {

if (helper.validateFarmerRecords(component, event)) {

helper.saveFarmerList(component, event);

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

// Validating and calling the save method from helper

readCSV: function(component, event, helper) {

const headers2 = new Array("Tipodeproducto", "Nombredeproducto", "Nombredecomprador", "Esunnuevocomprado", "Identificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].Tipodeproducto=="" ||

data1[k].Nombredeproducto==""||

data1[k].Nombredecomprador=="" ||

data1[k].Esunnuevocomprado=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var Tipodeproducto ;

var Nombredeproducto;

var Nombredecomprador;

var Esunnuevocomprado;

var Identificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

//alert(rows);

arr.pop();

//alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].Tipodeproducto=="" ||

data1[k].Nombredeproducto==""||

data1[k].Nombredecomprador=="" ||

data1[k].Esunnuevocomprado=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var Tipodeproducto ;

var Nombredeproducto;

var Nombredecomprador;

var Esunnuevocomprado;

var Identificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

// Saving records from uploading file .csv and .xlsx format

CreateContact : function (component,jsonstr){

// console.log('jsonstr' + jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

//alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

// alert('result callback>>'+ result);

// alert("Records Inserted Succesfully");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

addFarmerRecord: function(component, event) {

//get the Farmer List from component

var FarmerList = component.get("v.FarmerList");

//Add New Farmer Record

FarmerList.push({

'sobjectType': 'Farmers\_\_c',

'Edad\_\_c':'',

'Genero\_\_c':'',

'NOMBRE\_DE\_PRODUCTO\_\_c': '',

'TIPO\_DE\_PRODUCTO\_\_c':'',

'Name\_of\_Buyer\_\_c':'',

'Es\_esta\_nueva\_compradora\_\_c' : '',

'Contact\_Name\_\_c' : '',

});

component.set("v.FarmerList", FarmerList);

},

//checking Validation for all required fields

validateFarmerRecords: function(component, event) {

//Validate all account records

var isValid = true;

var FarmerList = component.get("v.FarmerList");

console.log('farmerList>>'+ FarmerList);

for (var i = 0; i < FarmerList.length; i++) {

if (FarmerList[i].Contact\_Name\_\_c == '' ||

//FarmerList[i].Edad\_\_c == '' ||

//FarmerList[i].Genero\_\_c == '' ||

FarmerList[i].NOMBRE\_DE\_PRODUCTO\_\_c == '' ||

FarmerList[i].TIPO\_DE\_PRODUCTO\_\_c == '' ||

FarmerList[i].Name\_of\_Buyer\_\_c == '' ||

FarmerList[i].Es\_esta\_nueva\_compradora\_\_c == '') {

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila "+(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

//Process the selected Farmers records to remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.FarmerList");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.FarmerList", finalFarmers);

},

// method to create records in server side

saveFarmerList: function(component, event, helper) {

//if (helper.validateFarmerRecords(component, event)) {

//Call Apex method and pass farmer list as a parameters

var action1 = component.get("c.saveCommoditySalesForm");

action1.setParams({

"Farmlist": component.get("v.FarmerList")

});

console.log('action>>'+ action1);

action1.setCallback(this, function(response) {

//get response status

var state = response.getState();

console.log('state>>'+state);

if (state === "SUCCESS") {

//set empty farmer list

component.set("v.FarmerList", []);

// alert('Registros guardados');//eng translation for alert:Records Saved Successfully

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '20000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action1);

// }

},

})

**Css:**

.THIS .slds-icon-utility-delete.slds-icon\_container{

padding-top: 14px;

padding-left: 18px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public class Commodity\_Sales\_Form\_Controller {

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list< fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Commodity Sales Form' limit 1];

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.TIPO\_DE\_PRODUCTO\_\_c = wrapper.Tipodeproducto;

far1.NOMBRE\_DE\_PRODUCTO\_\_c= wrapper.Nombredeproducto;

far1.Name\_of\_Buyer\_\_c = wrapper.Nombredecomprador;

far1.Es\_esta\_nueva\_compradora\_\_c = wrapper.Esunnuevocomprado;

if(wrapper.Identificacion!= null){

far1.Code\_\_c = wrapper.Identificacion;

}

ListtoInsert.add(far1);

}

if(ListtoInsert.size() > 0){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String Tipodeproducto;

public String Nombredeproducto;

public String Nombredecomprador;

public String Esunnuevocomprado;

public String Identificacion;

}

@AuraEnabled

public static void saveCommoditySalesForm(list<Farmers\_\_c> Farmlist){

system.debug('Farmlist>>'+Farmlist);

list<Farmers\_\_c> finalFarmlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Commodity Sales Form' limit 1];

for(Farmers\_\_c far : Farmlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmlist.add(far);

}

system.debug('finalFarmlist>>'+finalFarmlist);

Insert finalFarmlist;

}

@AuraEnabled

public static Map<String, String> getCommodity(){

Map<String, String> optionsCommodityMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.NOMBRE\_DE\_PRODUCTO\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsCommodityMap.put(ple.getValue(), ple.getLabel());

}

return optionsCommodityMap;

}

@AuraEnabled

public static Map<String, String> getTypeage(){

Map<String, String> optionsAgetypeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Edad\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsAgetypeMap.put(ple.getValue(), ple.getLabel());

}

return optionsAgetypeMap;

}

@AuraEnabled

public static Map<String, String> getGendertype(){

Map<String, String> optionsGenderMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Genero\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsGenderMap.put(ple.getValue(), ple.getLabel());

}

return optionsGenderMap;

}

@AuraEnabled

public static Map<String, String> getCommodityType(){

Map<String, String> optionsCommodityTypeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.TIPO\_DE\_PRODUCTO\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsCommodityTypeMap.put(ple.getValue(), ple.getLabel());

}

return optionsCommodityTypeMap;

}

@AuraEnabled

public static Map<String, String> getNewBuyer(){

Map<String, String> optionsNewBuyerMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Es\_esta\_nueva\_compradora\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsNewBuyerMap.put(ple.getValue(), ple.getLabel());

}

return optionsNewBuyerMap;

}

}

**TestClass:**

@isTest

public class Commodity\_Sales\_Form\_ControllerTest {

@istest

static void CommodityTest(){

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Commodity Sales Form' limit 1];

system.debug('RT@@'+RT);

Commodity\_Sales\_Form\_Controller ag = new Commodity\_Sales\_Form\_Controller();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

far1.NOMBRE\_DE\_PRODUCTO\_\_c='Tomates ';

far1.TIPO\_DE\_PRODUCTO\_\_c='Cultivo';

far1.Name\_of\_Buyer\_\_c ='test123';

far1.Es\_esta\_nueva\_compradora\_\_c='No';

far1.Contact\_Name\_\_c= con.id;

far1.RecordTypeId = RT.Id;

insert finalFarmlist1;

Commodity\_Sales\_Form\_Controller.saveCommoditySalesForm(finalFarmlist1);

Map<String, String> getCmdty = Commodity\_Sales\_Form\_Controller.getCommodity();

system.assertNotEquals(Null, getCmdty);

Map<String, String> getTypeAge = Commodity\_Sales\_Form\_Controller.getTypeage();

system.assertNotEquals(Null, getTypeAge);

Map<String, String> getGenderType = Commodity\_Sales\_Form\_Controller.getGendertype();

system.assertNotEquals(Null, getGenderType);

Map<String, String> getCmdtyType = Commodity\_Sales\_Form\_Controller.getCommodityType();

system.assertNotEquals(Null, getCmdtyType);

Map<String, String> getNewBuyer = Commodity\_Sales\_Form\_Controller.getNewBuyer();

system.assertNotEquals(Null, getNewBuyer);

}

@istest

static void CommodityCSVTest(){

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Commodity Sales Form' limit 1];

system.debug('RT@@'+RT);

Commodity\_Sales\_Form\_Controller ag = new Commodity\_Sales\_Form\_Controller();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

far2.NOMBRE\_DE\_PRODUCTO\_\_c='Tomates ';

far2.TIPO\_DE\_PRODUCTO\_\_c='Cultivo';

far2.Name\_of\_Buyer\_\_c ='test123';

far2.Es\_esta\_nueva\_compradora\_\_c='No';

far2.Contact\_Name\_\_c= con.id;

far2.RecordTypeId = RT.Id;

finalFarmlist1.add(far2);

insert finalFarmlist1;

Commodity\_Sales\_Form\_Controller.insertData(JSON.serialize(finalFarmlist1));

//Commodity\_Sales\_Form\_Controller.saveCommoditySalesForm(finalFarmlist1);

Map<String, String> getCmdty = Commodity\_Sales\_Form\_Controller.getCommodity();

system.assertNotEquals(Null, getCmdty);

Map<String, String> getTypeAge = Commodity\_Sales\_Form\_Controller.getTypeage();

system.assertNotEquals(Null, getTypeAge);

Map<String, String> getGenderType = Commodity\_Sales\_Form\_Controller.getGendertype();

system.assertNotEquals(Null, getGenderType);

Map<String, String> getCmdtyType = Commodity\_Sales\_Form\_Controller.getCommodityType();

system.assertNotEquals(Null, getCmdtyType);

Map<String, String> getNewBuyer = Commodity\_Sales\_Form\_Controller.getNewBuyer();

system.assertNotEquals(Null, getNewBuyer);

}

}

**MTC Training cmp:**

<aura:component controller="MTC\_TrainingController" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Attribute Declaration-->

<aura:attribute name="FarmerList" type="Farmers\_\_c[]"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!--Attribute Declaration for Picklists-->

<aura:attribute name="TrainingMap" type="Map"/>

<aura:attribute name="options\_TrainingMap" type="List" default="[]"/>

<aura:attribute name="IsnutritionMap" type="Map"/>

<aura:attribute name="options\_IsnutritionMap" type="List" default="[]"/>

<aura:attribute name="AgetypeMap" type="Map"/>

<aura:attribute name="options\_AgetypeMap" type="List" default="[]"/>

<aura:attribute name="GendertypeMap" type="Map"/>

<aura:attribute name="options\_GendertypeMap" type="List" default="[]"/>

<!--Attribute declaration for text field -->

<aura:attribute name="trainerState" type="String"/>

<aura:attribute name="trainingName" type="String"/>

<aura:attribute name="trainerName" type="String"/>

<!-- <aura:attribute name="trainerState" type="boolean" default="false"/>

<aura:attribute name="trainingName" type="boolean" default="false"/>

<aura:attribute name="trainerName" type="boolean" default="false"/> -->

<!--Attribute declaration for Lookup field-->

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<!-- <aura:handler name="change" value="{!v.selectedContactId}" action="{!c.handleContactChange}" /> -->

<aura:attribute name="selectedContactId" type="String" />

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<!-- Action Method/doInit-->

<aura:handler name="init" value="{!this}" action="{!c.doInit}"/>

<aura:handler name="init" value="{!this}" action="{!c.doNutrition}"/>

<!-- <aura:handler name="init" value="{!this}" action="{!c.doTypeofAge}"/>

<aura:handler name="init" value="{!this}" action="{!c.doTypeGender}"/>-->

<!-- Lightning components for designing UI -->

<div class="slds-m-around--x--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!-- <img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Times New Roman; color:green;">Formulario para participantes de capacitaciones en MTCs </span>

</div>

<lightning:layout>

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="center">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Nombre de entrenamiento<abbr class="slds-required" >\*</abbr></label>

<!-- <lightning:input label="Name of training"

value="{!v.trainingName}"

aura:id="trainingName"

required="true"

variant="label-hidden"/> -->

<lightning:input label="NameOfTraining" value="{!v.Name\_of\_training\_\_c}"

aura:id="TrainingName" required="true" onchange="{!c.captureTrainingName}" variant="label-hidden" class="message"/>

</div>

</lightning:layoutItem>

<!-- <lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="center">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Estado<abbr class="slds-required" >\*</abbr></label>

<lightning:input label="State"

value="{!v.trainerState}"

aura:id="training"

required="true"

variant="label-hidden"/>

<lightning:input label="State" value="{!v.State\_\_c}"

aura:id="state" required="true" onchange="{!c.captureState}" variant="label-hidden" class="message"/>

</div>

</lightning:layoutItem>-->

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="center">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Numero de identificación de entrenador<abbr class="slds-required" >\*</abbr></label>

<!-- <lightning:input label="Name of trainer"

value="{!v.trainerName}"

aura:id="trainer"

required="true"

variant="label-hidden"/> -->

<!--<lightning:input label="State" value="{!v.Name\_of\_Trainer\_\_c}"

aura:id="TrainerName" required="true" onchange="{!c.captureTrainerName}" variant="label-hidden" class="message"/>-->

<c:ContactTrainersLookup

objectAPIName="Contact"

label="Contact"

returnFields="['ID\_\_c']"

queryFields="['ID\_\_c']"

selectedId="{!v.selectedContactId}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z" />

</div>

</lightning:layoutItem>

</lightning:layout>

<!-- <lightning:layout>

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="center">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Nombre de entrenamiento<abbr class="slds-required" >\*</abbr></label>

<lightning:input label="Name of training"

value="{!v.trainingName}"

aura:id="trainingName"

required="true"

variant="label-hidden"/>

<lightning:input label="NameOfTraining" value="{!v.Name\_of\_training\_\_c}"

aura:id="TrainingName" required="true" onchange="{!c.captureTrainingName}" variant="label-hidden" class="message"/>

</div>

</lightning:layoutItem>

</lightning:layout>-->

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-truncate" >S.No</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="ID" align="center" >Identificación<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-cell-wrap" title="Age" align="center" >Edad<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Gender" align="center" >Genero<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<div class="slds-cell-wrap" title="Type of training " align="center">Tipo de entrenamiento <abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Is this a nutrition sensitive topic? " align="center">¿Es este un tema de nutrición? <abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Training date " align="center">fecha<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="Action" align="center">Action</div>

</th>-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.FarmerList}" var="Far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!--<lightning:input name="ID" type="text" value="{!Far.ID\_\_c}" />-->

<!-- <c:LookupPersonId objectAPIName="contact" IconName="standard:contact" />-->

<!-- Calling another component for creating lookup -->

<c:ContactFarmerLookUp

objectAPIName="Contact"

label="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId ="{!Far.Contact\_Name\_\_c}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<!-- <td>

<lightning:combobox name="Age" type="number" value="{!Far.Edad\_\_c}" options="{!v.options\_AgetypeMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="Gender" type="text" value="{!Far.Genero\_\_c}" options="{!v.options\_GendertypeMap}" placeholder="" variant="label-hidden"/>

</td>-->

<td>

<lightning:combobox name="Type of Training" type="text" value="{!Far.TIPO\_DE\_ENTRENAMIENTO\_\_c}" options="{!v.options\_TrainingMap}" placeholder="" variant="label-hidden" />

</td>

<td>

<lightning:combobox name="Is this a new buyer? Y/N " type="text" value="{!Far.Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c}" options="{!v.options\_IsnutritionMap}" placeholder="" variant="label-hidden" />

</td>

<td>

<!-- <lightning:input name="Training Date" type="date" value="{!Far.Training\_date\_\_c}" variant="label-hidden" class="date" dateStyle="short"/>-->

<lightning:input type="text" name="input" pattern="^((0[1-9])|([12][0-9])|(3[01]))\/((0[1-9])|(1[0-2]))\/((19[0-9]{2})|(2[0-9]{3}))$" required="true" value="{!Far.Training\_date\_\_c}" class="date" placeholder="DD/MM/AAAA" variant="label-hidden"/>

</td>

<!-- <td>

<a onclick="{!c.removeRow}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td> -->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 70rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario para participantes de capacitaciones en MTCs</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

</aura:component>

**Controller:**

({

// Showing the values in Popup while uploading files in both .csv and .xlsx format

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("Nombredeentrenamiento", "EntrenadorId", "Tipodeentrenamiento", "Esesteuntemadenutricion", "Fecha", "Identificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

// alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

// alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

//alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

// alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 6){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("Nombredeentrenamiento", "EntrenadorId", "Tipodeentrenamiento", "Esesteuntemadenutricion", "Fecha", "Identificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

//alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 6){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

//calling serverside/apex controller for getting PicklistValues

doInit : function(component, event, helper) {

helper.addFarmerRecord(component, event);

var action = component.get("c.getTypeofTraining");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TrainingMap = [];

var TrainingMapOptions = [];

for(var key in result){

TrainingMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TrainingMapOptions.push(state);

}

component.set("v.TrainingMap", TrainingMap);

component.set("v.options\_TrainingMap", TrainingMapOptions);

}

});

$A.enqueueAction(action);

},

doNutrition : function(component, event, helper) {

var action = component.get("c.getNutrition");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var NutritionMap = [];

var NutritionMapOptions = [];

for(var key in result){

NutritionMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

NutritionMapOptions.push(state);

}

component.set("v.IsnutritionMap", NutritionMap);

component.set("v.options\_IsnutritionMap", NutritionMapOptions);

}

});

$A.enqueueAction(action);

},

/\* doTypeofAge : function(component, event, helper) {

var action = component.get("c.getAgeMap");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var AgeMap = [];

var AgeMapOptions = [];

for(var key in result){

AgeMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

AgeMapOptions.push(state);

}

component.set("v.AgetypeMap", AgeMap);

component.set("v.options\_AgetypeMap", AgeMapOptions);

}

});

$A.enqueueAction(action);

},

doTypeGender : function(component, event, helper) {

var action = component.get("c.getTypeofGender");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var GenderMap = [];

var GenderMapOptions = [];

for(var key in result){

GenderMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

GenderMapOptions.push(state);

}

component.set("v.GendertypeMap", GenderMap);

component.set("v.options\_GendertypeMap", GenderMapOptions);

}

});

$A.enqueueAction(action);

},\*/

//Select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.FarmerList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//adding new row methods

addRow: function(component, event, helper) {

helper.addFarmerRecord(component, event);

},

// remove row method(single row)

removeRow: function(component, event, helper) {

//Get the Farmer list

var FarmerList = component.get("v.FarmerList");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

FarmerList.splice(index, 1);

component.set("v.FarmerList", FarmerList);

},

//final method to create records in server side

save: function(component, event, helper) {

var FarmerList = component.get("v.FarmerList");

// checking validation for required fields trainerState,trainingName,trainerName

//var Trstate =component.get("v.trainerState");

var TrngName =component.get("v.trainingName");

//var TnrName =component.get("v.trainerName");

var TnrName =component.get("v.selectedContactId");

// if(typeof Trstate == 'undefined' || typeof TrngName == 'undefined' ||

//typeof TnrName == 'undefined' ){

if( typeof TrngName == 'undefined' ||

typeof TnrName == 'undefined' ){

//alert('Las celdas obligatorias no pueden estar en blanco');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

else{

if (helper.validateFarmerRecords(component, event)) {

for(var i=0; i<FarmerList.length; i++){

//var Trstate =component.get("v.trainerState");

var TrngName =component.get("v.trainingName");

//var TnrName =component.get("v.trainerName");

var TnrName =component.get("v.selectedContactId");

// FarmerList[i].State\_\_c = Trstate;

FarmerList[i].Name\_of\_training\_\_c = TrngName;

FarmerList[i].Name\_of\_Trainer\_\_c = TnrName;

}

//Call Apex class and pass Farmer list parameters

var action1 = component.get("c.saveMTCTrainingForm");

action1.setParams({

"Farmerlist": component.get("v.FarmerList")

});

action1.setCallback(this, function(response) {

//get response status

var state = response.getState();

console.log('state>>'+state);

if (state === "SUCCESS") {

//set empty farmer list

component.set("v.FarmerList", []);

//alert('Registros guardados');//eng translation for alert:Records Saved Successfully

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action1);

}

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

//getting the values from clientside

captureState :function(component,event){

var State = event.getParam("value");

component.set("v.trainerState", State);

},

captureTrainerName :function(component,event){

var Trainer = event.getParam("value");

component.set("v.trainerName", Trainer);

},

captureTrainingName :function(component,event){

var Training = event.getParam("value");

component.set("v.trainingName", Training);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

// Validating and calling the save method from helper

readCSV: function(component, event, helper) {

const headers2 = new Array("Nombredeentrenamiento", "EntrenadorId", "Tipodeentrenamiento", "Esesteuntemadenutricion", "Fecha", "Identificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 6){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].Nombredeentrenamiento=="" ||

data1[k].EntrenadorId==""||

data1[k].Tipodeentrenamiento=="" ||

data1[k].Esesteuntemadenutricion=="" ||

data1[k].Fecha=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var Nombredeentrenamiento ;

var EntrenadorId;

var Tipodeentrenamiento;

var Esesteuntemadenutricion;

var Fecha;

var Identificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

//alert(rows);

arr.pop();

//alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 6){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].Nombredeentrenamiento=="" ||

data1[k].EntrenadorId==""||

data1[k].Tipodeentrenamiento=="" ||

data1[k].Esesteuntemadenutricion=="" ||

data1[k].Fecha=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var Nombredeentrenamiento ;

var EntrenadorId;

var Tipodeentrenamiento;

var Esesteuntemadenutricion;

var Fecha;

var Identificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

// Saving records from uploading file .csv and .xlsx format

CreateContact : function (component,jsonstr){

// console.log('jsonstr' + jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

//alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

// alert('result callback>>'+ result);

// alert("Records Inserted Succesfully");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

addFarmerRecord: function(component, event) {

//get the Farmer List from component

var FarmerList = component.get("v.FarmerList");

//Add New Farmer Record

FarmerList.push({

'sobjectType': 'Farmers\_\_c',

'ID\_\_c':'',

//'Edad\_\_c':'',

//'Genero\_\_c':'',

'State1\_\_c':'',

'TIPO\_DE\_ENTRENAMIENTO\_\_c': '',

'Training\_date\_\_c':'',

'Name\_of\_Trainer\_\_c':'',

'Name\_of\_training\_\_c':'',

'Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c':'',

'Contact\_Name\_\_c' : ''

});

component.set("v.FarmerList", FarmerList);

},

//Process the selected Farmers records to remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.FarmerList");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.FarmerList", finalFarmers);

},

//checking validation for all required fields

validateFarmerRecords: function(component, event) {

//Validate all account records

var isValid = true;

var farmersList = component.get("v.FarmerList");

for (var i = 0; i < farmersList.length; i++) {

if (farmersList[i].Contact\_Name\_\_c == '' ||

//farmersList[i].Edad\_\_c == '' ||

// farmersList[i].Genero\_\_c == '' ||

farmersList[i].Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c == '' ||

farmersList[i].TIPO\_DE\_ENTRENAMIENTO\_\_c == '' ||

farmersList[i].Training\_date\_\_c == '')

{

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila "+(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

})

**Css:**

.THIS .message .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .message .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"complete este campo";

}

.THIS .message .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .date .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .date .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"Utilice el siguiente formato: DD/MM/AAAA";

}

.THIS .date .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public class MTC\_TrainingController {

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list< fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC Training' limit 1];

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.Name\_of\_training\_\_c = wrapper.Nombredeentrenamiento;

far1.ExtenAgent\_\_c = wrapper.EntrenadorId;

far1.TIPO\_DE\_ENTRENAMIENTO\_\_c= wrapper.Tipodeentrenamiento;

far1.Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c = wrapper.Esesteuntemadenutricion;

far1.Training\_date\_\_c = wrapper.Fecha;

if(wrapper.Identificacion!= null){

far1.Code\_\_c = wrapper.Identificacion;

}

ListtoInsert.add(far1);

}

if(ListtoInsert.size() > 0){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String Nombredeentrenamiento;

public String EntrenadorId;

public String Tipodeentrenamiento;

public String Esesteuntemadenutricion;

public String Fecha;

public String Identificacion;

}

@AuraEnabled

public static void saveMTCTrainingForm(list<Farmers\_\_c> Farmerlist){

//Insert Farmerlist;

//012f0000000lbmUAAQ

system.debug('Farmlist>>'+Farmerlist);

list<Farmers\_\_c> finalFarmerlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC Training' limit 1];

// Id recordTypeId = Schema.SObjectType.Farmers\_\_c.getRecordTypeInfosByName().get('MTC Training Participant Form').getRecordTypeId();

for(Farmers\_\_c far : Farmerlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmerlist.add(far);

}

system.debug('finalFarmerlist>>'+finalFarmerlist);

Insert finalFarmerlist;

}

@AuraEnabled

public static Map<String, String> getTypeofTraining(){

Map<String, String> optionsTrainingMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.TIPO\_DE\_ENTRENAMIENTO\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsTrainingMap.put(ple.getValue(), ple.getLabel());

}

return optionsTrainingMap;

}

@AuraEnabled

public static Map<String, String> getAgeMap(){

Map<String, String> optionsAgeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Edad\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsAgeMap.put(ple.getValue(), ple.getLabel());

}

return optionsAgeMap;

}

@AuraEnabled

public static Map<String, String> getTypeofGender(){

Map<String, String> optionsGenderMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Genero\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsGenderMap.put(ple.getValue(), ple.getLabel());

}

return optionsGenderMap;

}

@AuraEnabled

public static Map<String, String> getNutrition(){

Map<String, String> optionsNutritionMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsNutritionMap.put(ple.getValue(), ple.getLabel());

}

return optionsNutritionMap;

}

}

**TestClass:**

@isTest

public class MTC\_TrainingControllerTest {

@istest

static void MTCtrainingTest(){

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC Training' limit 1];

system.debug('RT@@'+RT);

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

// far1.State1\_\_c='test';

far1.TIPO\_DE\_ENTRENAMIENTO\_\_c ='marketing';

far1.Training\_date\_\_c='12/03/2021';

far1.Name\_of\_Trainer\_\_c =con.Id;

far1.Name\_of\_training\_\_c='test';

far1.Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c='sí';

far1.Contact\_Name\_\_c= con.id;

far1.RecordTypeId = RT.Id;

insert finalFarmlist1;

MTC\_TrainingController.saveMTCTrainingForm(finalFarmlist1);

Map<String, String> getTypeofTraining = MTC\_TrainingController.getTypeofTraining();

system.assertNotEquals(Null, getTypeofTraining);

Map<String, String> getAgeMap = MTC\_TrainingController.getAgeMap();

system.assertNotEquals(Null, getAgeMap);

Map<String, String> getTypeofGender = MTC\_TrainingController.getTypeofGender();

system.assertNotEquals(Null, getTypeofGender);

Map<String, String> getNutrition = MTC\_TrainingController.getNutrition();

system.assertNotEquals(Null, getNutrition);

}

@istest

static void MTCtrainingCSVTest(){

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC Training' limit 1];

system.debug('RT@@'+RT);

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

// far1.State1\_\_c='test';

far2.TIPO\_DE\_ENTRENAMIENTO\_\_c ='marketing';

far2.Training\_date\_\_c='12/03/2021';

far2.Name\_of\_Trainer\_\_c =con.Id;

far2.Name\_of\_training\_\_c='test';

far2.Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c='sí';

far2.Contact\_Name\_\_c= con.id;

far2.RecordTypeId = RT.Id;

finalFarmlist1.add(far2);

insert finalFarmlist1;

MTC\_TrainingController.insertData(JSON.serialize(finalFarmlist1));

// MTC\_TrainingController.saveMTCTrainingForm(finalFarmlist1);

Map<String, String> getTypeofTraining = MTC\_TrainingController.getTypeofTraining();

system.assertNotEquals(Null, getTypeofTraining);

Map<String, String> getAgeMap = MTC\_TrainingController.getAgeMap();

system.assertNotEquals(Null, getAgeMap);

Map<String, String> getTypeofGender = MTC\_TrainingController.getTypeofGender();

system.assertNotEquals(Null, getTypeofGender);

Map<String, String> getNutrition = MTC\_TrainingController.getNutrition();

system.assertNotEquals(Null, getNutrition);

}

}

**MTC Business Grants cmp:**

<aura:component controller="MTC\_Business\_GrantController" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Attribute declaration-->

<aura:attribute name="FarmerList" type="Farmers\_\_c[]"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="selectedLookUpRecord" type="sObject" default="{}"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!-- Attribute declaration for picklist values-->

<aura:attribute name="BusinessMap" type="Map"/>

<aura:attribute name="options\_BusinessMap" type="List" default="[]"/>

<aura:attribute name="CategoryGrants" type="Map"/>

<aura:attribute name="options\_CategoryGrants" type="List" default="[]"/>

<aura:attribute name="AgeMap" type="Map"/>

<aura:attribute name="options\_AgeMap" type="List" default="[]"/>

<aura:attribute name="GenderTypeMap" type="Map"/>

<aura:attribute name="options\_GenderTypeMap" type="List" default="[]"/>

<!--Attribute declaration for lookup field-->

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<aura:handler name="change" value="{!v.selectedContactId}" action="{!c.handleContactChange}" />

<aura:attribute name="selectedContactId" type="String" />

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<!--Action Method/doInit-->

<aura:handler name="init" value="{!this}" action="{!c.doInit}"/>

<!-- <aura:handler name="init" value="{!this}" action="{!c.doAgeType}"/>

<aura:handler name="init" value="{!this}" action="{!c.doGender}"/>-->

<aura:handler name="init" value="{!this}" action="{!c.doCategory}"/>

<!--lightning components for designing UI-->

<div class="slds-m-around--x--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!-- <img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Times New Roman; color:green;">Formulario de subvención empresarial del MTC </span>

</div>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-truncate" >S.No</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="ID" align="center" >Nombre/numero de identificación<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-cell-wrap" title="Age" align="center" >Edad<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Gender" align="center" >Genero<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<div class="slds-cell-wrap" title="Received a Start up Business Grant" align="center">Recibieron subvención pequeña empresarial?<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Category of grant? One category for each of the training types " align="center">Categoría de subvención? (de acuerdo a las capacitaciones)<abbr class="slds-required" >\*</abbr>

</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Amount of the grant" align="center">Monto de la subvención<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="Action" align="center">Action</div>

</th>-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.FarmerList}" var="Far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!-- <lightning:input name="ID" type="text" value="{!Far.ID\_\_c}" />-->

<!-- <c:LookupPersonId objectAPIName="contact" IconName="standard:contact" selectedRecord="{!v.selectedLookUpRecord}"/>-->

<!-- calling another component for create lookup-->

<c:ContactFarmerLookUp

objectAPIName="Contact"

label="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId ="{!Far.Contact\_Name\_\_c}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<!-- <td>

<lightning:combobox name="Age" type="number" value="{!Far.Edad\_\_c}" options="{!v.options\_AgeMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="Gender" type="text" value="{!Far.Genero\_\_c}" options="{!v.options\_GenderTypeMap}" placeholder="" variant="label-hidden"/>

</td>-->

<td>

<lightning:combobox name="Received a Start up Business Grant" type="text" value="{!Far.RECIBIERON\_SUBVENCI\_N\_PEQUE\_A\_EMPRESARIA\_\_c}" options="{!v.options\_BusinessMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="Category of grant? One category for each of the training types " type="text" value="{!Far.Categor\_a\_de\_subvenci\_n\_\_c}" options="{!v.options\_CategoryGrants}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:input name="Amount of the grant" type="number" maxlength="16" value="{!Far.Amount\_of\_the\_grant\_\_c}" variant="label-hidden" class="message"/>

</td>

<!-- <td>

<a onclick="{!c.removeRow}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td> -->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas? </b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 70rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario de subvención empresarial del MTC</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

</aura:component>

**Controller:**

({

// Showing the values in Popup while uploading files in both .csv and .xlsx format

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("Recibieronsubvencionpequenaempresarial", "Categoriadesubvencion", "Montodelasubvencion", "Numerodeidentificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

// alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

//alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

// alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 4){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("Recibieronsubvencionpequenaempresarial", "Categoriadesubvencion", "Montodelasubvencion", "Numerodeidentificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

//alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 4){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

//calling server side/apex controller for getting picklist values

doInit : function(component, event, helper) {

helper.addFarmerRecord(component, event);

var action = component.get("c.getBusinessGrant");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var BusinessTypeMap = [];

var TypeBusinessMapOptions = [];

for(var key in result){

BusinessTypeMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeBusinessMapOptions.push(state);

}

component.set("v.BusinessMap",BusinessTypeMap);

component.set("v.options\_BusinessMap", TypeBusinessMapOptions);

}

});

$A.enqueueAction(action);

},

doCategory : function(component, event, helper) {

var action = component.get("c.getCategory");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var GrantCategoryMap = [];

var GrantCategoryMapOptions = [];

for(var key in result){

GrantCategoryMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

GrantCategoryMapOptions.push(state);

}

component.set("v.CategoryGrants",GrantCategoryMap);

component.set("v.options\_CategoryGrants", GrantCategoryMapOptions);

}

});

$A.enqueueAction(action);

},

//Select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.FarmerList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//add a new row method

addRow: function(component, event, helper) {

helper.addFarmerRecord(component, event);

},

//remove row method(single row)

removeRow: function(component, event, helper) {

//Get the Farmer list

var FarmerList = component.get("v.FarmerList");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

FarmerList.splice(index, 1);

component.set("v.FarmerList", FarmerList);

},

//final method to create record in serverside

save: function(component, event, helper) {

if (helper.validateFarmerRecords(component, event)){

helper.saveFarmerList(component, event, helper);

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

// Validating and Saving the records while uploading both .csv and .xlsx format

readCSV: function(component, event, helper) {

const headers2 = new Array("Recibieronsubvencionpequenaempresarial", "Categoriadesubvencion", "Montodelasubvencion", "Numerodeidentificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 4){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].Recibieronsubvencionpequenaempresarial=="" ||

data1[k].Categoriadesubvencion==""||

data1[k].Montodelasubvencion=="" ||

data1[k].Numerodeidentificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var Recibieronsubvencionpequenaempresarial ;

var Categoriadesubvencion;

var Montodelasubvencion;

var Numerodeidentificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

//alert(rows);

arr.pop();

//alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 4){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].Recibieronsubvencionpequenaempresarial=="" ||

data1[k].Categoriadesubvencion==""||

data1[k].Montodelasubvencion=="" ||

data1[k].Numerodeidentificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var Recibieronsubvencionpequenaempresarial ;

var Categoriadesubvencion;

var Montodelasubvencion;

var Numerodeidentificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

//Saving the records while uploading files

CreateContact : function (component,jsonstr){

// console.log('jsonstr' + jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

//alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

// alert('result callback>>'+ result);

// alert("Records Inserted Succesfully");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

addFarmerRecord: function(component, event) {

//get the Farmer List from component

var FarmerList = component.get("v.FarmerList");

//Add New Farmer Record

FarmerList.push({

'sobjectType': 'Farmers\_\_c',

'Categor\_a\_de\_subvenci\_n\_\_c':'',

'Edad\_\_c':'',

'Genero\_\_c':'',

'RECIBIERON\_SUBVENCI\_N\_PEQUE\_A\_EMPRESARIA\_\_c':'',

'Amount\_of\_the\_grant\_\_c': '',

'Contact\_Name\_\_c': ''

});

component.set("v.FarmerList", FarmerList);

},

//checking validation for all required fields

validateFarmerRecords: function(component, event) {

//Validate all account records

var isValid = true;

var farmersList = component.get("v.FarmerList");

console.log('farmersList'+farmersList);

for (var i = 0; i < farmersList.length; i++) {

if (farmersList[i].Contact\_Name\_\_c == '' ||

// farmersList[i].Edad\_\_c == '' ||

// farmersList[i].Genero\_\_c == '' ||

farmersList[i].RECIBIERON\_SUBVENCI\_N\_PEQUE\_A\_EMPRESARIA\_\_c == '' ||

farmersList[i].Categor\_a\_de\_subvenci\_n\_\_c == '' ||

farmersList[i].Amount\_of\_the\_grant\_\_c == '')

{

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila "+(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

//Process the selected Farmers record to remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.FarmerList");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.FarmerList", finalFarmers);

},

//final method to create records in server side

saveFarmerList: function(component, event, helper) {

//if (helper.validateFarmerRecords(component, event)) {

//Call Apex method and pass farmer list as a parameters

var acclook = component.get("v.selectedLookUpRecord");

//var accstring = component.get("v.selectedlookupstring");

if(acclook==null){

component.set('v.isSubmitted',true);

}

var action1 = component.get("c.saveMTCBusinessForm");

action1.setParams({

"Farmerlist": component.get("v.FarmerList")

/\*FarmerList.id\_\_c=null;

if(component.get("v.selectedLookUpRecord") != null && component.get("v.selectedLookUpRecord").Id != undefined){

FarmerList.ContactId = component.get("v.selectedLookUpRecord").Id;

//var sex = component.get("v.sex");

} \*/

});

action1.setCallback(this, function(response) {

//get response status

var state = response.getState();

console.log('state>>'+state);

if (state === "SUCCESS") {

//set empty farmer list

component.set("v.FarmerList", []);

//alert('Registros guardados');//eng translation for alert:Records Saved Successfully

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '20000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action1);

// }

},

})

**Css:**

.THIS .message .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .message .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"Ingrese un valor válido";

}

.THIS .message .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public class MTC\_Business\_GrantController {

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list< fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC Business Grant Form' limit 1];

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.RECIBIERON\_SUBVENCI\_N\_PEQUE\_A\_EMPRESARIA\_\_c= wrapper.Recibieronsubvencionpequenaempresarial;

// far1.Name\_of\_Trainer\_\_c = wrapper.NameofTrainer;

far1.Categor\_a\_de\_subvenci\_n\_\_c = wrapper.Categoriadesubvencion;

far1.Amount\_of\_the\_grant\_\_c = wrapper.Montodelasubvencion;

if(wrapper.Numerodeidentificacion!= null){

far1.Code\_\_c = wrapper.Numerodeidentificacion;

}

ListtoInsert.add(far1);

}

if(ListtoInsert.size() > 0){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String Recibieronsubvencionpequenaempresarial;

//public String NameofTrainer;

public String Categoriadesubvencion;

public Decimal Montodelasubvencion;

public String Numerodeidentificacion;

}

@AuraEnabled

public static void saveMTCBusinessForm(list<Farmers\_\_c> Farmerlist){

system.debug('Farmlist>>'+Farmerlist);

list<Farmers\_\_c> finalFarmerlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC Business Grant Form' limit 1];

for(Farmers\_\_c far : Farmerlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmerlist.add(far);

}

system.debug('finalFarmerlist>>'+finalFarmerlist);

Insert finalFarmerlist;

}

@AuraEnabled

public static Map<String, String> getBusinessGrant(){

Map<String, String> optionsBusinessGrantMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.RECIBIERON\_SUBVENCI\_N\_PEQUE\_A\_EMPRESARIA\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsBusinessGrantMap.put(ple.getValue(), ple.getLabel());

}

return optionsBusinessGrantMap;

}

@AuraEnabled

public static Map<String, String> getCategory(){

Map<String, String> optionsCGrantMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Categor\_a\_de\_subvenci\_n\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsCGrantMap.put(ple.getValue(), ple.getLabel());

}

return optionsCGrantMap;

}

@AuraEnabled

public static Map<String, String> getAgetypeMap(){

Map<String, String> optionsAgetypeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Edad\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsAgetypeMap.put(ple.getValue(), ple.getLabel());

}

return optionsAgetypeMap;

}

@AuraEnabled

public static Map<String, String> getGendertype(){

Map<String, String> optionsGenderMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Genero\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsGenderMap.put(ple.getValue(), ple.getLabel());

}

return optionsGenderMap;

}

}

**TestClass:**

@isTest

public class MTC\_Business\_GrantControllerTest {

@istest

static void MTCBussinessGrantTest(){

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC Business Grant Form' limit 1];

system.debug('RT@@'+RT);

Ag\_Training\_Participant ag = new Ag\_Training\_Participant();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

far1.RECIBIERON\_SUBVENCI\_N\_PEQUE\_A\_EMPRESARIA\_\_c='No';

far1.Categor\_a\_de\_subvenci\_n\_\_c ='marketing';

far1.Amount\_of\_the\_grant\_\_c=452;

far1.Contact\_Name\_\_c= con.id;

far1.RecordTypeId = RT.Id;

insert finalFarmlist1;

MTC\_Business\_GrantController.saveMTCBusinessForm(finalFarmlist1);

Map<String, String> getGrant = MTC\_Business\_GrantController.getBusinessGrant();

system.assertNotEquals(Null, getGrant);

Map<String, String> getCategory = MTC\_Business\_GrantController.getCategory();

system.assertNotEquals(Null, getCategory);

Map<String, String> getAge = MTC\_Business\_GrantController.getAgetypeMap();

system.assertNotEquals(Null, getAge);

Map<String, String> getGender = MTC\_Business\_GrantController.getGendertype();

system.assertNotEquals(Null, getGender);

}

@istest

static void MTCBussinessGrantCSVTest(){

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC Business Grant Form' limit 1];

system.debug('RT@@'+RT);

MTC\_Business\_GrantController ag = new MTC\_Business\_GrantController();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

far2.RECIBIERON\_SUBVENCI\_N\_PEQUE\_A\_EMPRESARIA\_\_c='No';

far2.Categor\_a\_de\_subvenci\_n\_\_c ='marketing';

far2.Amount\_of\_the\_grant\_\_c=452;

far2.Contact\_Name\_\_c= con.id;

far2.RecordTypeId = RT.Id;

finalFarmlist1.add(far2);

insert finalFarmlist1;

//MTC\_Business\_GrantController.saveMTCBusinessForm(finalFarmlist1);

MTC\_Business\_GrantController.insertData(JSON.serialize(finalFarmlist1));

Map<String, String> getGrant = MTC\_Business\_GrantController.getBusinessGrant();

system.assertNotEquals(Null, getGrant);

Map<String, String> getCategory = MTC\_Business\_GrantController.getCategory();

system.assertNotEquals(Null, getCategory);

Map<String, String> getAge = MTC\_Business\_GrantController.getAgetypeMap();

system.assertNotEquals(Null, getAge);

Map<String, String> getGender = MTC\_Business\_GrantController.getGendertype();

system.assertNotEquals(Null, getGender);

}

}

**MTC Business Started cmp:**

<aura:component controller="MTC\_Business\_StartedController" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Attribute declaration-->

<aura:attribute name="FarmerList" type="Farmers\_\_c[]"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!--Attribute declaration for picklist values-->

<aura:attribute name="AgeTypeMap" type="Map"/>

<aura:attribute name="options\_AgeTypeMap" type="List" default="[]"/>

<aura:attribute name="TypeofGenderMap" type="Map"/>

<aura:attribute name="options\_TypeofGenderMap" type="List" default="[]"/>

<aura:attribute name="startKitsMap" type="Map"/>

<aura:attribute name="options\_StartkitsMap" type="List" default="[]"/>

<aura:attribute name="startedPoultryMap" type="Map"/>

<aura:attribute name="options\_StartedpoultryMap" type="List" default="[]"/>

<aura:attribute name="startupHomeMap" type="Map"/>

<aura:attribute name="options\_StartupHomeMap" type="List" default="[]"/>

<aura:attribute name="startedHomeMap" type="Map"/>

<aura:attribute name="options\_StartedhomeMap" type="List" default="[]"/>

<!--Attribute declaration for lookup field-->

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<aura:handler name="change" value="{!v.selectedContactId}" action="{!c.handleContactChange}" />

<aura:attribute name="selectedContactId" type="String" />

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<!--Action Method/doInit-->

<!-- <aura:handler name="init" value="{!this}" action="{!c.doInit}"/>

<aura:handler name="init" value="{!this}" action="{!c.doGender}"/>-->

<aura:handler name="init" value="{!this}" action="{!c.doPoultry}"/>

<aura:handler name="init" value="{!this}" action="{!c.doStartedpolutry}"/>

<aura:handler name="init" value="{!this}" action="{!c.doHomeGarden}"/>

<aura:handler name="init" value="{!this}" action="{!c.doStartedgarden}"/>

<!--lightning components for designing UI-->

<div class="slds-m-around--x--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!-- <img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Times New Roman; color:green;">Empresas iniciadas de MTC (aves de corral o jardinería doméstica) y kits de inicio proporcionados </span>

</div>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-cell-wrap" align="center" >S.No</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="ID" align="center" >Nombre/ numero de identificación<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-cell-wrap" title="Age" align="center" >Edad<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Gender" align="center" >Genero<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<div class="slds-cell-wrap" title="Received Start Up Kits For Poultry Business" align="center"> ¿Recibieron kits de inicio para negocio avícola?<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Started Poultry Business " align="center">¿Iniciaron un negocio avícola? <abbr class="slds-required" >\*</abbr>

</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Received Startup kits For Home Gardening Business" align="center">¿Recibieron kits de inicio para negocio de jardinería doméstica?<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Started Home Gardening Business" align="center">¿Iniciaron un negocio de jardinería doméstica?<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="Action" align="center">Action</div>

</th>-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.FarmerList}" var="Far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!--<lightning:input name="ID" type="text" value="{!Far.ID\_\_c}" />-->

<!-- <c:LookupPersonId objectAPIName="contact" IconName="standard:contact" />-->

<!--Calling another component for creating lookup-->

<c:ContactFarmerLookUp

objectAPIName="Contact"

label="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId ="{!Far.Contact\_Name\_\_c}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<!-- <td>

<lightning:combobox name="Age" type="number" value="{!Far.Edad\_\_c}" options="{!v.options\_AgeTypeMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="Gender" type="text" value="{!Far.Genero\_\_c}" options="{!v.options\_TypeofGenderMap}" placeholder="" variant="label-hidden"/>

</td>-->

<td>

<lightning:combobox name="Received Start Up kits For Poultry Business" type="text" value="{!Far.Recibieron\_kits\_de\_inicio\_para\_negocio\_\_c}" options="{!v.options\_StartkitsMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="Started Poultry Business" type="text" value="{!Far.QUE\_INICIARON\_EL\_NEGOCIO\_AV\_COLA\_\_c}" options="{!v.options\_StartedpoultryMap}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox name="Received Startup kits for Home Gardening Business" type="text" value="{!Far.KITS\_DE\_INICIO\_PARA\_NEGOCIO\_DE\_JARDINER\_\_c}" options="{!v.options\_StartupHomeMap}" placeholder="" variant="label-hidden" />

</td>

<td>

<lightning:combobox name="Started Home Gardening Business" type="text" value="{!Far.QUE\_INICIARON\_UN\_NEGOCIO\_DE\_JARDINER\_A\_D\_\_c}" options="{!v.options\_StartedhomeMap}" placeholder="" variant="label-hidden" />

</td>

<!-- <td>

<a onclick="{!c.removeRow}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td> -->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 70rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Empresas iniciadas de MTC (aves de corral o jardinería doméstica) y kits de inicio proporcionados</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

</aura:component>

**Controller:**

({

// Showing the values in Popup while uploading files in both .csv and .xlsx format

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("Recibieronkits", "Iniciaronunnegocioavicola", "Inicioparanegociodejardineria", "Negociodejardineriadomestica", "Numerodeidentificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

// alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

//alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

// alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("Recibieronkits", "Iniciaronunnegocioavicola", "Inicioparanegociodejardineria", "Negociodejardineriadomestica", "Numerodeidentificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

//alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

//Calling serverside/apex controller for getting picklist values

doPoultry : function(component, event, helper) {

helper.addFarmerRecord(component, event);

var action = component.get("c.getstartPolutry");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var ReceivedPoultryMap = [];

var ReceivedPoultrymapOptions = [];

for(var key in result){

ReceivedPoultryMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

ReceivedPoultrymapOptions.push(state);

}

component.set("v.startKitsMap",ReceivedPoultryMap);

component.set("v.options\_StartkitsMap", ReceivedPoultrymapOptions);

}

});

$A.enqueueAction(action);

},

doStartedpolutry : function(component, event, helper) {

var action = component.get("c.getStartedbusiness");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var PoultryMap = [];

var PoultryMapOptions = [];

for(var key in result){

PoultryMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

PoultryMapOptions.push(state);

}

component.set("v.startedPoultryMap",PoultryMap);

component.set("v.options\_StartedpoultryMap", PoultryMapOptions);

}

});

$A.enqueueAction(action);

},

doHomeGarden : function(component, event, helper) {

var action = component.get("c.getReceivedgardening");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var HomeGardeningMap = [];

var HomeGardeningMapOptions = [];

for(var key in result){

HomeGardeningMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

HomeGardeningMapOptions.push(state);

}

component.set("v.startupHomeMap",HomeGardeningMap);

component.set("v.options\_StartupHomeMap", HomeGardeningMapOptions);

}

});

$A.enqueueAction(action);

},

doStartedgarden : function(component, event, helper) {

var action = component.get("c.getStartedHome");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var GardenMap = [];

var GardenMapOptions = [];

for(var key in result){

GardenMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

GardenMapOptions.push(state);

}

component.set("v.startedHomeMap",GardenMap);

component.set("v.options\_StartedhomeMap", GardenMapOptions);

}

});

$A.enqueueAction(action);

},

//select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.FarmerList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//add row method

addRow: function(component, event, helper) {

helper.addFarmerRecord(component, event);

},

//remove row method(single row)

removeRow: function(component, event, helper) {

//Get the Farmer list

var FarmerList = component.get("v.FarmerList");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

FarmerList.splice(index, 1);

component.set("v.FarmerList", FarmerList);

},

//final method for create record in serverside

save: function(component, event, helper) {

if (helper.validateFarmerRecords(component, event)) {

helper.saveFarmerList(component, event, helper);

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

// Validating and Saving the records while uploading both .csv and .xlsx format

readCSV: function(component, event, helper) {

const headers2 = new Array("Recibieronkits", "Iniciaronunnegocioavicola", "Inicioparanegociodejardineria", "Negociodejardineriadomestica", "Numerodeidentificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].Recibieronkits=="" ||

data1[k].Iniciaronunnegocioavicola==""||

data1[k].Inicioparanegociodejardineria=="" ||

data1[k].Negociodejardineriadomestica=="" ||

data1[k].Numerodeidentificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var Recibieronkits;

var Iniciaronunnegocioavicola;

var Inicioparanegociodejardineria;

var Negociodejardineriadomestica;

var Numerodeidentificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

//alert(rows);

arr.pop();

//alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 5){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].Recibieronkits=="" ||

data1[k].Iniciaronunnegocioavicola==""||

data1[k].Inicioparanegociodejardineria=="" ||

data1[k].Negociodejardineriadomestica=="" ||

data1[k].Numerodeidentificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var Recibieronkits;

var Iniciaronunnegocioavicola;

var Inicioparanegociodejardineria;

var Negociodejardineriadomestica;

var Numerodeidentificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

//Saving the records while uploading files

CreateContact : function (component,jsonstr){

// console.log('jsonstr' + jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

//alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

// alert('result callback>>'+ result);

// alert("Records Inserted Succesfully");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

addFarmerRecord: function(component, event) {

//get the Farmer List from component

var FarmerList = component.get("v.FarmerList");

//Add New Farmer Record

FarmerList.push({

'sobjectType': 'Farmers\_\_c',

'ID\_\_c':'',

'Edad\_\_c':'',

'Genero\_\_c':'',

'Recibieron\_kits\_de\_inicio\_para\_negocio\_\_c':'',

'QUE\_INICIARON\_EL\_NEGOCIO\_AV\_COLA\_\_c': '',

'KITS\_DE\_INICIO\_PARA\_NEGOCIO\_DE\_JARDINER\_\_c':'',

'QUE\_INICIARON\_UN\_NEGOCIO\_DE\_JARDINER\_A\_D\_\_c':'',

'Contact\_Name\_\_c' : ''

});

component.set("v.FarmerList", FarmerList);

},

//checking Validation for all required field

validateFarmerRecords: function(component, event) {

//Validate all account records

var isValid = true;

var FarmerList = component.get("v.FarmerList");

console.log('farmerList>>'+ FarmerList);

for (var i = 0; i < FarmerList.length; i++) {

console.log('FarmerList[i].ID\_\_c>>'+ FarmerList[i].ID\_\_c);

console.log('FarmerList[i].Received\_startup\_kits\_\_c>>'+FarmerList[i].Received\_startup\_kits\_\_c);

console.log('FarmerList[i].Started\_home\_gardening\_business\_\_c'+ FarmerList[i].Started\_home\_gardening\_business\_\_c);

console.log(' FarmerList[i].Started\_Poultry\_business\_\_c'+ FarmerList[i].Started\_Poultry\_business\_\_c);

console.log(' FarmerList[i].received\_Start\_up\_kits\_for\_Poultry\_bus\_\_c'+ FarmerList[i].received\_Start\_up\_kits\_for\_Poultry\_bus\_\_c);

if (FarmerList[i].Contact\_Name\_\_c== '' ||

//FarmerList[i].Edad\_\_c== '' ||

//FarmerList[i].Genero\_\_c== '' ||

FarmerList[i].Recibieron\_kits\_de\_inicio\_para\_negocio\_\_c == '' ||

FarmerList[i].QUE\_INICIARON\_EL\_NEGOCIO\_AV\_COLA\_\_c == '' ||

FarmerList[i].KITS\_DE\_INICIO\_PARA\_NEGOCIO\_DE\_JARDINER\_\_c == '' ||

FarmerList[i].QUE\_INICIARON\_UN\_NEGOCIO\_DE\_JARDINER\_A\_D\_\_c == '') {

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila "+(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

//Process the selected Farmers record to remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.FarmerList");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.FarmerList", finalFarmers);

},

//final method to create records in server side

saveFarmerList: function(component, event, helper) {

//if (helper.validateFarmerRecords(component, event)) {

//Call Apex method and pass farmer list as a parameters

var action1 = component.get("c.saveMTCBusinessStartedForm");

action1.setParams({

"Farmerlist": component.get("v.FarmerList")

});

action1.setCallback(this, function(response) {

//get response status

var state = response.getState();

console.log('state>>'+state);

if (state === "SUCCESS") {

//set empty farmer list

component.set("v.FarmerList", []);

// alert('Registros guardados'); //eng translation for alert:Records Saved Successfully

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action1);

// }

},

})

**Css:**

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public class MTC\_Business\_StartedController {

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list< fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC business started and startup kits' limit 1];

system.debug('Success');

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.Recibieron\_kits\_de\_inicio\_para\_negocio\_\_c= wrapper.Recibieronkits;

// far1.Name\_of\_Trainer\_\_c = wrapper.NameofTrainer;

far1.QUE\_INICIARON\_EL\_NEGOCIO\_AV\_COLA\_\_c = wrapper.Iniciaronunnegocioavicola;

far1.KITS\_DE\_INICIO\_PARA\_NEGOCIO\_DE\_JARDINER\_\_c = wrapper.Inicioparanegociodejardineria;

far1.QUE\_INICIARON\_UN\_NEGOCIO\_DE\_JARDINER\_A\_D\_\_c = wrapper.Negociodejardineriadomestica;

//if(wrapper.numerodeidentificacion!= null){

far1.Code\_\_c = wrapper.Numerodeidentificacion;

//}

ListtoInsert.add(far1);

}

if(ListtoInsert.size() > 0){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String Recibieronkits;

//public String NameofTrainer;

public String Iniciaronunnegocioavicola;

public String Inicioparanegociodejardineria;

public String Negociodejardineriadomestica;

public String Numerodeidentificacion;

}

@AuraEnabled

public static void saveMTCBusinessStartedForm(list<Farmers\_\_c> Farmerlist){

//Insert Farmerlist;

system.debug('Farmlist>>'+Farmerlist);

list<Farmers\_\_c> finalFarmerlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC business started and startup kits' limit 1];

for(Farmers\_\_c far : Farmerlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmerlist.add(far);

}

system.debug('finalFarmerlist>>'+finalFarmerlist);

Insert finalFarmerlist;

}

@AuraEnabled

public static Map<String, String> TypeofAge(){

Map<String, String> optionsAgetypeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Edad\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsAgetypeMap.put(ple.getValue(), ple.getLabel());

}

return optionsAgetypeMap;

}

@AuraEnabled

public static Map<String, String> TypeofGender(){

Map<String, String> optionsGenderMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Genero\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsGenderMap.put(ple.getValue(), ple.getLabel());

}

return optionsGenderMap;

}

@AuraEnabled

public static Map<String, String> getstartPolutry(){

Map<String, String> optionsPolutryMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Recibieron\_kits\_de\_inicio\_para\_negocio\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsPolutryMap.put(ple.getValue(), ple.getLabel());

}

return optionsPolutryMap;

}

@AuraEnabled

public static Map<String, String> getStartedbusiness(){

Map<String, String> optionsBusinessMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.QUE\_INICIARON\_EL\_NEGOCIO\_AV\_COLA\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsBusinessMap.put(ple.getValue(), ple.getLabel());

}

return optionsBusinessMap;

}

@AuraEnabled

public static Map<String, String> getReceivedgardening(){

Map<String, String> optionsHomeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.KITS\_DE\_INICIO\_PARA\_NEGOCIO\_DE\_JARDINER\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsHomeMap.put(ple.getValue(), ple.getLabel());

}

return optionsHomeMap;

}

@AuraEnabled

public static Map<String, String> getStartedHome(){

Map<String, String> optionsStartedhomeMap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.QUE\_INICIARON\_UN\_NEGOCIO\_DE\_JARDINER\_A\_D\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsStartedhomeMap.put(ple.getValue(), ple.getLabel());

}

return optionsStartedhomeMap;

}

}

**TestClass:**

@isTest

public class MTC\_Business\_StartedControllerTest {

@istest

static void AgMTCBussStartedTest(){

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC business started and startup kits' limit 1];

system.debug('RT@@'+RT);

MTC\_Business\_StartedController ag = new MTC\_Business\_StartedController();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

far1.Recibieron\_kits\_de\_inicio\_para\_negocio\_\_c='sí';

far1.QUE\_INICIARON\_EL\_NEGOCIO\_AV\_COLA\_\_c ='sí';

far1.KITS\_DE\_INICIO\_PARA\_NEGOCIO\_DE\_JARDINER\_\_c='sí';

far1.QUE\_INICIARON\_UN\_NEGOCIO\_DE\_JARDINER\_A\_D\_\_c='sí';

far1.Contact\_Name\_\_c= con.id;

far1.RecordTypeId = RT.Id;

insert finalFarmlist1;

MTC\_Business\_StartedController.saveMTCBusinessStartedForm(finalFarmlist1);

Map<String, String> getstartPolutry = MTC\_Business\_StartedController.getstartPolutry();

system.assertNotEquals(Null, getstartPolutry);

Map<String, String> getAge = MTC\_Business\_StartedController.TypeofAge();

system.assertNotEquals(Null, getAge);

Map<String, String> getGender = MTC\_Business\_StartedController.TypeofGender();

system.assertNotEquals(Null, getGender);

Map<String, String> getStartedbusiness = MTC\_Business\_StartedController.getStartedbusiness();

system.assertNotEquals(Null, getStartedbusiness);

Map<String, String> getReceivedgardening = MTC\_Business\_StartedController.getReceivedgardening();

system.assertNotEquals(Null, getReceivedgardening);

Map<String, String> getStartedHome = MTC\_Business\_StartedController.getStartedHome();

system.assertNotEquals(Null, getStartedHome);

}

@istest

static void AgMTCBussStartedCSVTest(){

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'MTC business started and startup kits' limit 1];

system.debug('RT@@'+RT);

MTC\_Business\_StartedController ag = new MTC\_Business\_StartedController();

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

far2.Recibieron\_kits\_de\_inicio\_para\_negocio\_\_c='sí';

far2.QUE\_INICIARON\_EL\_NEGOCIO\_AV\_COLA\_\_c ='sí';

far2.KITS\_DE\_INICIO\_PARA\_NEGOCIO\_DE\_JARDINER\_\_c='sí';

far2.QUE\_INICIARON\_UN\_NEGOCIO\_DE\_JARDINER\_A\_D\_\_c='sí';

far2.Contact\_Name\_\_c= con.id;

far2.RecordTypeId = RT.Id;

finalFarmlist1.add(far2);

insert finalFarmlist1;

// MTC\_Business\_StartedController.saveMTCBusinessStartedForm(finalFarmlist1);

MTC\_Business\_StartedController.insertData(JSON.serialize(finalFarmlist1));

Map<String, String> getstartPolutry = MTC\_Business\_StartedController.getstartPolutry();

system.assertNotEquals(Null, getstartPolutry);

Map<String, String> getAge = MTC\_Business\_StartedController.TypeofAge();

system.assertNotEquals(Null, getAge);

Map<String, String> getGender = MTC\_Business\_StartedController.TypeofGender();

system.assertNotEquals(Null, getGender);

Map<String, String> getStartedbusiness = MTC\_Business\_StartedController.getStartedbusiness();

system.assertNotEquals(Null, getStartedbusiness);

Map<String, String> getReceivedgardening = MTC\_Business\_StartedController.getReceivedgardening();

system.assertNotEquals(Null, getReceivedgardening);

Map<String, String> getStartedHome = MTC\_Business\_StartedController.getStartedHome();

system.assertNotEquals(Null, getStartedHome);

}

}

**Youth Nutrition cmp:**

<aura:component controller="Youth\_Nutrition\_Training\_Controller" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<!--Attribute declaration-->

<aura:attribute name="farmersList" type="Farmers\_\_c[]"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!--Attribute declaration for picklistValues-->

<aura:attribute name="AgetypeMap" type="Map"/>

<aura:attribute name="option\_AgetypeMap" type="List" default="[]"/>

<aura:attribute name="GendertypeMap" type="Map"/>

<aura:attribute name="option\_GendertypeMap" type="List" default="[]"/>

<!--Attribute declaration for text field-->

<aura:attribute name="toggleValue" type="boolean" default="false"/>

<aura:attribute name="trainingYouth" type="boolean" default="false"/>

<aura:attribute name="trainerYouth" type="boolean" default="false"/>

<aura:attribute name="trainingDate" type="boolean" default="false"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<!--Attribute declaration for lookup field-->

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<!-- <aura:handler name="change" value="{!v.selectedContactId}" action="{!c.handleContactChange}" /> -->

<aura:attribute name="selectedContactId" type="String" />

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<!--Attribute declaration for picklist-->

<aura:attribute name="NSTYMap" type="Map"/>

<aura:attribute name="options\_NSTY" type="List" default="[]"/>

<aura:attribute name="ValueNSTY" type="String"/>

<!--Action Method/doInIt-->

<aura:handler name="init" value="{!this}" action="{!c.addRow}"/>

<aura:handler name="init" value="{!this}" action="{!c.doNSTY}"/>

<!-- <aura:handler name="init" value="{!this}" action="{!c.doAgeMap}"/>

<aura:handler name="init" value="{!this}" action="{!c.doGenderMap}"/>-->

<!--lightning component for designing UI-->

<div class="slds-m-around--x--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Times New Roman; color:green;"> Formulario de capacitación en nutrición para jóvenes</span>

</div>

<lightning:layout>

<lightning:layoutItem size="6">

<!-- <div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Nombre de capacitación<abbr class="slds-required" >\*</abbr></label>

<lightning:input label="Name of training"

value="{!v.trainingYouth}"

aura:id="training"

required="true"

variant="label-hidden" class="message"/>

</div>-->

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Nombre de entrenador<abbr class="slds-required" >\*</abbr></label>

<lightning:input label="Name of Trainer"

value="{!v.trainerYouth}"

aura:id="trainer"

required="true"

variant="label-hidden" class="message"/>

<!-- <c:ContactTrainersLookup

objectAPIName="Contact"

label="Contact"

returnFields="['ID\_\_c']"

queryFields="['ID\_\_c']"

selectedId="{!v.selectedContactId}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z" /> -->

</div>

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Fecha<abbr class="slds-required" >\*</abbr></label>

<lightning:input type="text" name="input" pattern="^((0[1-9])|([12][0-9])|(3[01]))\/((0[1-9])|(1[0-2]))\/((19[0-9]{2})|(2[0-9]{3}))$" required="true" value="{!v.trainingDate}" class="date" placeholder="DD/MM/AAAA" variant="label-hidden"/>

</div>

</lightning:layoutItem>

<lightning:layoutItem size="6">

<!-- <div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">fecha<abbr class="slds-required" >\*</abbr></label>-->

<!-- <lightning:input label="Training Date"

type="date"

value="{!v.trainingDate}"

aura:id="trainingDateId"

required="true"

variant="label-hidden"

class="date" dateStyle="short"/>-->

<!-- <lightning:input type="text" name="input" pattern="^((0[1-9])|([12][0-9])|(3[01]))\/((0[1-9])|(1[0-2]))\/((19[0-9]{2})|(2[0-9]{3}))$" required="true" value="{!v.trainingDate}" class="date" placeholder="DD/MM/AAAA" variant="label-hidden"/>

</div>-->

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Es este un tema de nutrición?<abbr class="slds-required" >\*</abbr></label>

<lightning:combobox label="Is this a nutrition sensitive topic"

value="{!v.Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c}"

aura:id="Nutrition"

class="message"

variant="label-hidden"

placeholder=""

options="{!v.options\_NSTY}"

onchange="{!c.CaptureNSTY}" />

</div>

</lightning:layoutItem>

</lightning:layout>

</div>

<div class="slds-float\_right slds-p-bottom\_small">

<!-- <div class="slds-float\_right slds-p-bottom\_small slds-p-left\_x-small">-->

<lightning:button label="Agregar Fila"

iconName="utility:add"

iconPosition="left"

variant="brand"

onclick="{!c.addRow}"/>

<lightning:button label="Quitar Fila"

iconName="utility:delete"

iconPosition="left"

variant="destructive"

onclick="{!c.handleConfirmDialog}"/>

<!-- New code for upload button-->

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-cell-wrap" align="center">S.No</div>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Last name/Id">Apellido/Identificación<abbr class="slds-required" >\*</abbr></div></center>

</th>

<!-- <th scope="col">

<center><div class="slds-cell-wrap" title="Age">Edad<abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Gender">Genero<abbr class="slds-required" >\*</abbr></div></center>

</th>-->

<th scope="col">

<center><div class="slds-cell-wrap" title="Training name">Nombre de capacitación<abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Pretest score">Resultado antes de capacitación<abbr class="slds-required" ></abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Pretest score possible">Resultado máximo posible<abbr class="slds-required" >\*</abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Post test score">Resultado después de capacitación<abbr class="slds-required" ></abbr></div></center>

</th>

<th scope="col">

<center><div class="slds-cell-wrap" title="Post test score possible">Resultado máximo posible<abbr class="slds-required" >\*</abbr></div></center>

</th>

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.farmersList}" var="far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}" />

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!-- <lightning:input name="farName" type="text" maxlength="50" value="{!far.Contact\_Name\_\_c}"/>-->

<!-- <c:LookupPersonId objectAPIName="contact" IconName="standard:contact" />-->

<!--calling another component for create lookup-->

<c:ContactFarmerLookUp

objectAPIName="Contact"

label="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId="{!far.Contact\_Name\_\_c}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<!-- <td>

<lightning:combobox name="Age"

value="{!far.Edad\_\_c}"

options="{!v.option\_AgetypeMap}"

variant="label-hidden"

placeholder=""

/>

</td>

<td>

<lightning:combobox name="Gender"

value="{!far.Genero\_\_c}"

options="{!v.option\_GendertypeMap}"

variant="label-hidden"

placeholder=""

/>

</td>-->

<td>

<lightning:input name="farNameTrn" type="text" maxlength="50" value="{!far.Name\_of\_training\_\_c}" variant="label-hidden"/>

</td>

<td>

<lightning:input name="farPreSc"

type="number"

maxlength="50"

value="{!far.Pretest\_score\_\_c}"

variant="label-hidden"

/>

</td>

<td>

<lightning:input name="farPreScPo" type="number" maxlength="50" value="{!far.Pretest\_score\_possible\_\_c}" variant="label-hidden"/>

</td>

<td>

<lightning:input name="farPoSco" type="number" maxlength="50" value="{!far.Post\_test\_score\_\_c}" variant="label-hidden"/>

</td>

<td>

<lightning:input name="farPoScoPo" type="number" maxlength="50" value="{!far.Post\_test\_score\_possible\_\_c}" variant="label-hidden"/>

</td>

</tr>

</aura:iteration>

</tbody>

</table>

</div>

</div>

</div>

</div>

</div>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.saveFarmers}" />

</div>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 90rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario de capacitación en nutrición para jóvenes</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

<!--<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>-->

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

<!-- New code for popup box confirmation-->

</aura:component>

**Controller:**

({

//calling serverside/apex controller for getting picklist values

doAgeMap : function(component, event, helper) {

helper.addFarmerRecord(component, event);

var action = component.get("c.AgeMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var AgeMaptype = [];

var TypeAgeOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

AgeMaptype.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeAgeOptions.push(state);

}

component.set("v.AgetypeMap", AgeMaptype);

component.set("v.option\_AgetypeMap", TypeAgeOptions);

}

});

$A.enqueueAction(action);

},

doNSTY : function(component, event, helper) {

var action = component.get("c.NutritionSensitiveMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var NSTYMap = [];

var NSTYOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

NSTYMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

NSTYOptions.push(state);

}

component.set("v.NSTYMap", NSTYMap);

component.set("v.options\_NSTY", NSTYOptions);

}

});

$A.enqueueAction(action);

},

doGenderMap : function(component, event, helper) {

var action = component.get("c.getGenderMap");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeGenderMap = [];

var TypeGenderOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

TypeGenderMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeGenderOptions.push(state);

}

component.set("v.GendertypeMap", TypeGenderMap);

component.set("v.option\_GendertypeMap", TypeGenderOptions);

}

});

$A.enqueueAction(action);

},

// add new row method

addRow : function(component, event, helper) {

var farmersList = component.get("v.farmersList");

farmersList.push({

'sobjectType': 'Farmers\_\_c',

'Name': '',

'ID\_\_c': '',

'Age\_\_c':'',

'Gender\_\_c':'',

'Name\_of\_training\_\_c': '',

'Pretest\_score\_\_c': '',

'Pretest\_score\_possible\_\_c': '',

'Post\_test\_score\_\_c': '',

'Contact\_Name\_\_c':'',

'Post\_test\_score\_possible\_\_c': '',

'Name\_of\_training\_Youth\_\_c':'',

'Name\_of\_trainer\_Youth\_\_c':'',

'Training\_date\_\_c':null,

'Is\_this\_nutrition\_sensitive\_topic\_Youth\_\_c':''

});

component.set("v.farmersList", farmersList);

},

/\* addRow: function(component, event, helper) {

helper.addFarmerRecord(component, event);

},\*/

//select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.farmersList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

//final method to create records in server side

saveFarmers: function(component, event, helper) {

console.log('Inside Savefarmers JS-1');

var farmersList = component.get("v.farmersList");

//var TrainingYouth = component.get("v.trainingYouth");

var trainerYouth = component.get("v.trainerYouth");

var TrainerDate = component.get("v.trainingDate");

var ValueNSTY = component.get("v.ValueNSTY");

// Validating Records

if( trainerYouth == false || TrainerDate == false ||

typeof ValueNSTY == 'undefined'){

//alert('Las celdas obligatorias no pueden estar en blanco');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

else{

if (helper.validateFarmerRecords(component, event)) {

var ValueNSTY = component.get("v.ValueNSTY");

// Adding Type of Trainer and Trainer all values adding to FarmerList

for(var i=0; i<farmersList.length; i++){

// farmersList[i].Name\_of\_training\_Youth\_\_c = component.get("v.trainingYouth");

farmersList[i].Name\_of\_trainer\_Youth\_\_c = component.get("v.trainerYouth");

farmersList[i].Training\_date\_\_c = component.get("v.trainingDate");

farmersList[i].Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c = ValueNSTY;

/\* if(component.get("v.toggleValue")){

farmersList[i].Is\_this\_nutrition\_sensitive\_topic\_Youth\_\_c = 'Yes';

}else{

farmersList[i].Is\_this\_nutrition\_sensitive\_topic\_Youth\_\_c = 'No';

} \*/

}

//Call Apex method and pass farmer list as a parameters

var action = component.get("c.saveFarmerRecords");

action.setParams({

"farmerList": component.get("v.farmersList")

});

action.setCallback(this, function(response) {

//get response status

var state = response.getState();

console.log('State'+state);

if (state === "SUCCESS") {

//set empty farmer list

component.set("v.farmersList", []);

//alert('Registros guardados');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action);

}

}

},

//getting value from clientside

CaptureNSTY :function(component,event){

var NSTYValue = event.getParam("value");

//alert(TOTRValue);

component.set("v.ValueNSTY", NSTYValue);

},

/\* new code for both csv and xlsx\*/

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("Nombredeentrenador" , "Nombrecapacitacion" , "Resultadoantesdecapacitacion" , "Resultadomaximoposible" , "Resultadodespuesdecapacitacion" , "Resultadomaximopossible" , "Fecha" , "Esesteuntemadenutricion" , "Identificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

// alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

//alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

// alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("Nombredeentrenador" , "Nombrecapacitacion" , "Resultadoantesdecapacitacion" , "Resultadomaximoposible" , "Resultadodespuesdecapacitacion" , "Resultadomaximopossible" , "Fecha" , "Esesteuntemadenutricion" , "Identificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

//alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

readCSV: function(component, event, helper) {

const headers2 = new Array("Nombredeentrenador" , "Nombrecapacitacion" , "Resultadoantesdecapacitacion" , "Resultadomaximoposible" , "Resultadodespuesdecapacitacion" , "Resultadomaximopossible" , "Fecha" , "Esesteuntemadenutricion" , "Identificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

var st = "sí";

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].Nombredeentrenador=="" ||

data1[k].Nombrecapacitacion==""||

// data1[k].Nombredecapacitacion=="" ||

data1[k].Resultadomaximoposible=="" ||

data1[k].Resultadomaximopossible=="" ||

data1[k].Fecha=="" ||

data1[k].Esesteuntemadenutricion=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

//else if(data1[k].Esesteuntemadenutricion =='sí' && data1[k].Resultadoantesdecapacitacion=="" || data1[k].Resultadodespuesdecapacitacion=="" ){

else if(data1[k].Esesteuntemadenutricion ==st && data1[k].Resultadoantesdecapacitacion=="" ) {

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor complete Resultado antes de capacitación y Resultado después de capacitación " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

else if(data1[k].Esesteuntemadenutricion ==st && data1[k].Resultadodespuesdecapacitacion=="") {

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor complete Resultado antes de capacitación y Resultado después de capacitación " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var Nombredeentrenador ;

var Nombrecapacitacion;

var Resultadoantesdecapacitacion;

var Resultadomaximoposible;

var Resultadodespuesdecapacitacion;

var Resultadomaximopossible;

var Fecha;

var Esesteuntemadenutricion;

var Identificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

//alert(rows);

arr.pop();

//alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 9){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows ;

var st = "sí";

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].Nombredeentrenador=="" ||

data1[k].Nombrecapacitacion==""||

//data1[k].Nombredecapacitacion=="" ||

data1[k].Resultadomaximoposible=="" ||

data1[k].Resultadomaximopossible=="" ||

data1[k].Fecha=="" ||

data1[k].Esesteuntemadenutricion=="" ||

data1[k].Identificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

//else if(data1[k].Esesteuntemadenutricion =='sí' && data1[k].Resultadoantesdecapacitacion=="" ){

else if(data1[k].Esesteuntemadenutricion ==st && data1[k].Resultadoantesdecapacitacion=="") {

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor complete Resultado antes de capacitación y Resultado después de capacitación " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

else if(data1[k].Esesteuntemadenutricion ==st && data1[k].Resultadodespuesdecapacitacion=="") {

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor complete Resultado antes de capacitación y Resultado después de capacitación " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var Nombredeentrenador ;

var Nombrecapacitacion;

var Resultadoantesdecapacitacion;

var Resultadomaximoposible;

var Resultadodespuesdecapacitacion;

var Resultadomaximopossible;

var Fecha;

var Esesteuntemadenutricion;

var Identificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

})

**Helper:**

({

/\*addFarmerRecord : function(component, event, helper) {

var farmersList = component.get("v.farmersList");

farmersList.push({

'sobjectType': 'Farmers\_\_c',

'Name': '',

'ID\_\_c': '',

'Edad\_\_c':'',

'Genero\_\_c':'',

'Name\_of\_training\_\_c': '',

'Pretest\_score\_\_c': '',

'Pretest\_score\_possible\_\_c': '',

'Post\_test\_score\_\_c': '',

'Post\_test\_score\_possible\_\_c': '',

'Name\_of\_training\_Youth\_\_c':'',

'Name\_of\_trainer\_Youth\_\_c':'',

'Training\_date\_\_c':null,

'Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c':''

});

component.set("v.farmersList", farmersList);

},\*/

//Process the selected Farmers records to remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.farmersList");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.farmersList", finalFarmers);

},

//checking validation for pretest and posttestscore field is not empty when nutrition type is yes/sí

validateFarmerRecords: function(component, event, helper) {

//Validate all account records

var isValid = true;

var farmersList = component.get("v.farmersList");

var checkvalue = component.get("v.ValueNSTY");

//console.log('checkvalue--'+checkvalue);

if (checkvalue == 'sí'){

for (var i = 0; i < farmersList.length; i++) {

console.log('farmersList[i].Pretest\_score\_\_c>>'+farmersList[i].Pretest\_score\_\_c);

if (farmersList[i].Pretest\_score\_\_c == '' ||

farmersList[i].Post\_test\_score\_\_c == '' ||

farmersList[i].Name\_of\_training\_\_c == null ||

//farmersList[i].Duplicate\_ID\_\_c == null ||

farmersList[i].Pretest\_score\_possible\_\_c == null ||

farmersList[i].Post\_test\_score\_possible\_\_c == null ) {

isValid = false;

//alert('por favor complete Resultado antes de capacitación y Resultado después de capacitación'+ (i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor complete Resultado antes de capacitación y Resultado después de capacitación",

duration:' 3000',

key: 'info\_alt',

type: 'error',

mode: 'pester'

});

toastEvent.fire();

}

}

//checking validation for all required fields when nutrition type is no

}else{

for (var i = 0; i < farmersList.length; i++) {

console.log('else>>');

if( farmersList[i].Name\_of\_training\_\_c == '' ||

// farmersList[i].Edad\_\_c == '' ||

//farmersList[i].Genero\_\_c == '' ||

farmersList[i].Contact\_Name\_\_c == '' ||

farmersList[i].Pretest\_score\_possible\_\_c == null ||

farmersList[i].Post\_test\_score\_possible\_\_c ==null ){

//alert('Las celdas obligatorias no pueden estar en blanco en la fila'+' '+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila "+(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

}

for (var i = 0; i < farmersList.length; i++) {

console.log('farmersList[i].Pretest\_score\_\_c>>'+farmersList[i].Pretest\_score\_\_c);

if (farmersList[i].Contact\_Name\_\_c == '' ||

//farmersList[i].Post\_test\_score\_\_c == '' ||

//farmersList[i].Name\_of\_training\_\_c == null ||

farmersList[i].Post\_test\_score\_possible\_\_c == '' ||

farmersList[i].Pretest\_score\_possible\_\_c == '' ||

farmersList[i].Name\_of\_training\_\_c == null ) {

isValid = false;

// alert('Las celdas obligatorias no pueden estar en blanco en la fila'+(i+1));

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila "+(i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

CreateContact : function (component,jsonstr){

// console.log('jsonstr' + jsonstr);

// alert('jsonstr>>>>>' + jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

// alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

// alert('result callback>>'+ result);

//alert("Records Inserted Succesfully");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

})

**Css:**

.THIS .date .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .date .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"Utilice el siguiente formato: DD/MM/AAAA";

}

.THIS .date .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .message .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .message .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"complete este campo";

}

.THIS .message .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS .button {

background-color: #04AA6D;

border: 1px ;

margin: 25px 50px 75px 100px;

color: white;

padding: 10px;

text-align: left;

text-decoration: none;

display: inline-block;

font-size: 10px;

margin: 4px 2px;

cursor: pointer;

}

.THIS.input {

width: 12ch;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public class Youth\_Nutrition\_Training\_Controller {

@AuraEnabled

public static void saveFarmerRecords(List<Farmers\_\_c> farmerList){

system.debug('Inside Save Farmer Server CAll'+farmerList);

//Insert farmerList;

list<Farmers\_\_c> finalFarmlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Youth Nutrition Training Form' limit 1];

for(Farmers\_\_c far : farmerlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmlist.add(far);

}

system.debug('finalFarmlist>>'+finalFarmlist);

Insert finalFarmlist;

}

@AuraEnabled

public static Map<String, String> NutritionSensitiveMethod(){

Map<String, String> optionsNSTY = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsNSTY.put(ple.getValue(), ple.getLabel());

}

return optionsNSTY;

}

@AuraEnabled

public static Map<String, String> AgeMethod(){

Map<String, String> optionstypeAge = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Edad\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionstypeAge.put(ple.getValue(), ple.getLabel());

}

return optionstypeAge;

}

@AuraEnabled

public static Map<String, String> getGenderMap(){

Map<String, String> optionsGendertype = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Genero\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsGendertype.put(ple.getValue(), ple.getLabel());

}

return optionsGendertype;

}

/\* new code for upload csv files by using wrapper for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list<fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Youth Nutrition Training Form' limit 1];

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.Name\_of\_trainer\_Youth\_\_c = wrapper.Nombredeentrenador;

//far1.Name\_of\_training\_Youth\_\_c = wrapper.Nombrecapacitacion;

far1.Name\_of\_training\_\_c = wrapper.Nombrecapacitacion;

if(wrapper.Resultadoantesdecapacitacion=='' || wrapper.Resultadoantesdecapacitacion == null){

far1.Pretest\_Score\_\_c = 0.00;

}else{

far1.Pretest\_Score\_\_c = Decimal.valueOf(wrapper.Resultadoantesdecapacitacion);

}

if(wrapper.Resultadomaximoposible=='' || wrapper.Resultadomaximoposible == null){

far1.Pretest\_Score\_Possible\_\_c = 0.00;

}else{

far1.Pretest\_Score\_Possible\_\_c = Decimal.valueOf(wrapper.Resultadomaximoposible);

}

if(wrapper.Resultadodespuesdecapacitacion=='' || wrapper.Resultadodespuesdecapacitacion == null){

far1.Post\_test\_score\_\_c = 0.00;

}else{

far1.Post\_test\_score\_\_c = Decimal.valueOf(wrapper.Resultadodespuesdecapacitacion);

}

if(wrapper.Resultadomaximopossible=='' || wrapper.Resultadomaximopossible == null){

far1.Post\_test\_score\_\_c = 0.00;

}else{

far1.Post\_test\_score\_possible\_\_c = Decimal.valueOf(wrapper.Resultadomaximopossible);

}

far1.Training\_date\_\_c = wrapper.Fecha;

far1.Es\_este\_un\_tema\_sensible\_a\_la\_nutrici\_n\_\_c = wrapper.Esesteuntemadenutricion;

if(wrapper.Identificacion!= null){

far1.Code\_\_c = wrapper.Identificacion;

}

System.Debug(wrapper.Esesteuntemadenutricion);

ListtoInsert.add(far1);

System.debug('ListtoInsert>>>>>'+ListtoInsert);

}

if(ListtoInsert.size() > 0){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

System.Debug('ListtoInsert>>>>+>>>>+>>>'+ListtoInsert);

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String Nombredeentrenador;

public String Nombrecapacitacion;

// public String Nombredecapacitacion;

public String Resultadoantesdecapacitacion;

public String Resultadomaximoposible;

public String Resultadodespuesdecapacitacion;

public String Resultadomaximopossible;

public String Fecha;

public String Esesteuntemadenutricion;

public String Identificacion;

}

}

**TestClass:**

@isTest

public class Youth\_Nutrition\_Training\_ControllerTest {

@istest

static void YouthNutriTest(){

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Youth Nutrition Training Form' limit 1];

system.debug('RT@@'+RT);

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

far1.Age\_\_c='<30';

far1.Gender\_\_c ='Male';

far1.Name\_of\_training\_\_c='test';

far1.Pretest\_score\_\_c=45;

far1.Pretest\_score\_possible\_\_c=54;

far1.Post\_test\_score\_\_c =65;

far1.Post\_test\_score\_possible\_\_c=76;

far1.Name\_of\_training\_Youth\_\_c='test';

far1.Name\_of\_trainer\_Youth\_\_c='Semillas';

far1.Training\_date\_\_c ='12/03/2021';

far1.Is\_this\_nutrition\_sensitive\_topic\_Youth\_\_c='Yes';

far1.Contact\_Name\_\_c= con.id;

far1.RecordTypeId = RT.Id;

insert finalFarmlist1;

Youth\_Nutrition\_Training\_Controller.saveFarmerRecords(finalFarmlist1);

Map<String, String> NutritionSensitiveMethod = Youth\_Nutrition\_Training\_Controller.NutritionSensitiveMethod();

system.assertNotEquals(Null, NutritionSensitiveMethod);

Map<String, String> AgeMethod = Youth\_Nutrition\_Training\_Controller.AgeMethod();

system.assertNotEquals(Null, AgeMethod);

Map<String, String> getGenderMap = Youth\_Nutrition\_Training\_Controller.getGenderMap();

system.assertNotEquals(Null, getGenderMap);

}

@istest

static string YouthNutriCSVTest(){

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test1234 name';

con.email='test1234@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Youth Nutrition Training Form' limit 1];

system.debug('RT@@'+RT);

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

far2.Age\_\_c='<30';

far2.Gender\_\_c ='Male';

far2.Name\_of\_training\_\_c='test';

far2.Pretest\_score\_\_c=45;

far2.Pretest\_score\_possible\_\_c=54;

far2.Post\_test\_score\_\_c =65;

far2.Post\_test\_score\_possible\_\_c=76;

far2.Name\_of\_training\_Youth\_\_c='test';

far2.Name\_of\_trainer\_Youth\_\_c='Semillas';

far2.Training\_date\_\_c ='12/03/2021';

far2.Is\_this\_nutrition\_sensitive\_topic\_Youth\_\_c='Yes';

far2.Contact\_Name\_\_c= con.id;

far2.RecordTypeId = RT.Id;

finalFarmlist1.add(far2);

insert finalFarmlist1;

// Youth\_Nutrition\_Training\_Controller.insertData('far2');

Youth\_Nutrition\_Training\_Controller.insertData(JSON.serialize(finalFarmlist1));

//List<Youth\_Nutrition\_Training\_Controller.fieldWrapper> wrapperList = Youth\_Nutrition\_Training\_Controller.insertData(finalFarmlist1);

Map<String, String> NutritionSensitiveMethod = Youth\_Nutrition\_Training\_Controller.NutritionSensitiveMethod();

system.assertNotEquals(Null, NutritionSensitiveMethod);

Map<String, String> AgeMethod = Youth\_Nutrition\_Training\_Controller.AgeMethod();

system.assertNotEquals(Null, AgeMethod);

Map<String, String> getGenderMap = Youth\_Nutrition\_Training\_Controller.getGenderMap();

system.assertNotEquals(Null, getGenderMap);

return returnresponse;

}

}

**Extension Agent cmp:**

<aura:component controller="Extension\_Agent\_ParticipantController" implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome,force:hasRecordId,forceCommunity:availableForAllPageTypes,force:lightningQuickAction" access="global" >

<!-- Loads XLSX js libary -->

<ltng:require scripts="{!$Resource.XL}"/>

<aura:attribute name="far" type="Farmers\_\_c" default="{'sObjectType' : 'Farmers\_\_c'}"/>

<aura:attribute name="FarmerList" type="Farmers\_\_c[]"/>

<aura:attribute name="isSelectAll" type="boolean" default="false"/>

<aura:attribute name="showConfirmDialog" type="boolean" default="false"/>

<!--Attribute declaration for lookup field-->

<aura:attribute name="PersonID" type="list" />

<aura:attribute name="issearching" type="Boolean" default="false"/>

<aura:attribute name="farmerId" type="string" />

<!-- <aura:handler name="change" value="{!v.selectedContactId}" action="{!c.handleContactChange}" /> -->

<aura:attribute name="selectedContactId" type="String" />

<aura:attribute name="con" type="contact" default="{'sObjectType' : 'Contact'}"/>

<aura:attribute name="TypeOfGenderMap" type="Map"/>

<aura:attribute name="options\_TypeOfGender" type="List" default="[]"/>

<aura:attribute name="TypeOfNRTMap" type="Map"/>

<aura:attribute name="options\_TypeOfNRT" type="List" default="[]"/>

<aura:attribute name="TypeOfAgeMap" type="Map"/>

<aura:attribute name="options\_TypeOfAge" type="List" default="[]"/>

<aura:attribute name="TypeOfTOTMap" type="Map"/>

<aura:attribute name="options\_TypeOfTOT" type="List" default="[]"/>

<!-- <aura:attribute name="NameOTG" type="String"/>-->

<aura:attribute name="NameTR" type="String"/>

<aura:attribute name="IsNutrition" type="String"/>

<aura:attribute name="TRDate" type="String"/>

<aura:attribute name="options" type="List" default="[{'label': 'File Upload', 'value': 'FileUpload'}]"/>

<aura:attribute name="value" type="String" default="FileUpload"/>

<aura:attribute name="jsonstr" type="String" default=""/>

<aura:handler name="init" value="{!this}" action="{!c.doInit}"/>

<!-- <aura:handler name="init" value="{!this}" action="{!c.doTypeOfGender}"/>

<aura:handler name="init" value="{!this}" action="{!c.doAgetype}"/>-->

<aura:handler name="init" value="{!this}" action="{!c.doTypeOfTraining}"/>

<!-- Import attributes start -->

<aura:attribute name="ShowModule" type="boolean" default="false"/>

<aura:attribute name="showcard" type="boolean" default="false"/>

<aura:attribute name="closeModal" type="boolean" default="false"/>

<!-- Import attributes end -->

<div class="slds-m-around--x--large" align="center">

<div class="slds-container--x--large slds-box slds-theme\_default">

<div class="slds-form--stacked">

<div class="slds-form-element">

<div class="slds-p-bottom\_large" align="center">

<!-- <img src="{!$Resource.POA\_Logo}" width="500"/>-->

<div class="slds-p-top\_large">

<span style="font-weight:bold; font-size:28px; font-family: Times New Roman; color:green;">Formulario para extensionistas capacitados </span>

</div>

<lightning:layout>

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Es un tema relacionado a la nutrición?<abbr class="slds-required" >\*</abbr> </label>

<lightning:combobox label="Is this a nutrition related topic?" value="{!v.Es\_este\_un\_tema\_relacionado\_con\_la\_nutr\_\_c}"

options="{!v.options\_TypeOfNRT}" aura:id="field" required="true" onchange="{!c.captureNutri}" variant="label-hidden" placeholder="" class="message" />

</div>

</lightning:layoutItem>

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_small slds-p-bottom\_small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">fecha<abbr class="slds-required" >\*</abbr></label>

<!-- <lightning:input label="Training Date" type="Date" value="{!v.Training\_Date\_extension\_\_c}" onchange="{!c.captureDate}" placeholder="" variant="label-hidden" class="stateFieldSize"/>-->

<lightning:input type="text" name="input" pattern="^((0[1-9])|([12][0-9])|(3[01]))\/((0[1-9])|(1[0-2]))\/((19[0-9]{2})|(2[0-9]{3}))$" required="true" onchange="{!c.captureDate}" class="date" placeholder="DD/MM/AAAA" variant="label-hidden" />

</div>

</lightning:layoutItem>

</lightning:layout>

<lightning:layout>

<!-- <lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_small slds-p-bottom\_small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Nombre de capacitación</label>

<lightning:input label="Name of Training" required="true" type="text" value="{!v.Name\_of\_training\_\_c}" placeholder="" variant="label-hidden" class="stateFieldSize"/>

</div>

</lightning:layoutItem>-->

<lightning:layoutItem size="6">

<div class="slds-form-element\_\_control slds-p-left\_small slds-p-right\_small slds-p-top\_x-small" align="left">

<label style="font-size:12px; font-weight:bold; font-family: Arial;">Nombre de entrenador<abbr class="slds-required" >\*</abbr></label>

<!-- <lightning:input label="Trainer Name" value="{!v.Trainer\_name\_\_c}"

aura:id="field" onchange="{!c.captureNOTR}" required="true" variant="label-hidden" class="message" />-->

<c:ContactTrainersLookup

objectAPIName="Contact"

label="Contact"

returnFields="['ID\_\_c']"

queryFields="['ID\_\_c']"

selectedId="{!v.selectedContactId}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z" />

</div>

</lightning:layoutItem>

</lightning:layout>

<!-- <aura:if isTrue = "{!v.closeModal}">-->

<div class="slds-float\_right slds-p-bottom\_small">

<lightning:button label="Agregar Fila" iconName="utility:add" iconPosition="left" variant="brand" onclick="{!c.addRow}" />

<lightning:button label="Quitar Fila" iconName="utility:delete" iconPosition="left" variant="destructive" onclick="{!c.handleConfirmDialog}" />

<label for="file-upload" class="custom-file-upload" style="

color: #fff;

padding: 7.8px 15px 9.2px 15px !important;

/\* border: 1px solid #b5b5b5; \*/

border-bottom-color: #7f7f7f;

background: green;

font-weight: bold;

font-size: 1em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

margin-left: 5px;

margin-top: 1px;

"> Subir Archivo

</label>

<input id="file-upload" type="file" aura:id="file" onchange="{!c.showexceldata}" />

</div>

<div class="container-fluid">

<table class="slds-table slds-table\_bordered slds-table\_cell-buffer">

<thead>

<tr class="slds-text-title\_caps">

<th scope="col">

<label class="slds-checkbox">

<ui:inputCheckbox value="{!v.isSelectAll}" change="{!c.handleSelectAll}" aura:id="selectAll"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<th scope="col">

<div class="slds-cell-wrap" >S.No</div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="ID" align="center" >Nombre/ numero de identificacion<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-cell-wrap" title="Age" align="center" >Edad<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Gender" align="center" >Genero<abbr class="slds-required" >\*</abbr></div>

</th>-->

<th scope="col">

<div class="slds-cell-wrap" title="Name of training" align="center">Nombre de capacitación<abbr class="slds-required" >\*</abbr></div>

</th>

<th scope="col">

<div class="slds-cell-wrap" title="Type of training" align="center">Tipo de entrenamiento<abbr class="slds-required" >\*</abbr></div>

</th>

<!-- <th scope="col">

<div class="slds-truncate" title="Action" align="center">Action</div>

</th>-->

</tr>

</thead>

<tbody>

<aura:iteration items="{!v.FarmerList}" var="Far" indexVar="index">

<tr>

<th>

<label class="slds-checkbox">

<ui:inputCheckbox aura:id="checkFarmer" value="" text="{!index}"/>

<span class="slds-checkbox--faux" />

<span class="slds-form-element\_\_label"></span>

</label>

</th>

<td style="padding-top: 23px;">

{!index + 1}

</td>

<td>

<!-- <lightning:input name="ID" type="text" value="{!Far.ID\_\_c}" variant="label-hidden"/>-->

<!-- <c:ContactFarmerLookUp

objectAPIName="Contact"

label="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId ="{!Far.Contact\_Name\_\_c}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>-->

<c:ContactFarmerLookUp

objectAPIName="Contact"

label="Contact"

returnFields="['Project\_Participant\_Code\_\_c']"

queryFields="['Project\_Participant\_Code\_\_c']"

selectedId ="{!Far.Contact\_Name\_\_c}"

variant="label-hidden"

filter="CreatedDate > 2001-01-01T00:00:01Z"/>

</td>

<!-- <td>

<lightning:combobox aura:id="Age" label="Age" value="{!Far.Edad\_\_c}" options="{!v.options\_TypeOfAge}" placeholder="" variant="label-hidden"/>

</td>

<td>

<lightning:combobox aura:id="Gender" label="Gender" value="{!Far.Genero\_\_c}" options="{!v.options\_TypeOfGender}" placeholder="" variant="label-hidden"/>

</td>-->

<td>

<lightning:input name="Name of training " type="text" value="{!Far.Name\_of\_training\_\_c}" variant="label-hidden"/>

</td>

<td>

<lightning:combobox aura:id="TOT" label="Type Of Training" value="{!Far.Tipo\_de\_entrenamiento\_extension\_\_c}" options="{!v.options\_TypeOfTOT}" placeholder="" variant="label-hidden"/>

</td>

<!-- <td>

<a onclick="{!c.removeRow}" data-record="{!index}">

<lightning:icon iconName="utility:delete" size="small" alternativeText="Delete"/>

<span class="slds-assistive-text">Delete</span>

</a>

</td> -->

</tr>

</aura:iteration>

</tbody>

</table>

<div class="slds-align\_absolute-center slds-p-top\_small">

<lightning:button variant="brand" label="Guardar" title="Brand action" onclick="{!c.save}" />

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<!--Confirmation Pop-Up section Start-->

<div class="slds-m-around\_xx-large">

<aura:if isTrue="{!v.showConfirmDialog}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<h1 class="slds-text-heading--medium">confirmación</h1>

</header>

<!--Modal Box Header End-->

<!--Modal Box Content Start-->

<div class="slds-modal\_\_content slds-p-around--medium">

<center><b>Esta seguro que quiere eliminar estas filas?</b></center>

</div>

<!--Modal Box Content End-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>

<lightning:button name='No' label='No' onclick='{!c.handleConfirmDialogNo}'/>

<!-- <lightning:button variant="brand" name='Yes' label='sí' onclick='{!c.handleConfirmDialogYes}'/>-->

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

<aura:if isTrue = "{!v.closeModal}">

<aura:if isTrue = "{!v.showcard}">

<!--Modal Box Start-->

<div role="dialog" class="slds-modal slds-fade-in-open ">

<div class="slds-modal\_\_container">

<!--Modal Box Header Start-->

<header class="slds-modal\_\_header">

<aura:html tag="style">

.slds-modal\_\_container{

width: 100% !important;

max-width: 70rem !important;

}

</aura:html>

<lightning:button iconName="utility:close"

onclick="{! c.closeModel }"

variant="bare-inverse"

class="slds-modal\_\_close"/>

<h1 class="slds-text-heading--medium">Formulario para extensionistas capacitados</h1>

</header>

<!--Modal Box Header End-->

<!-- <aura:if isTrue="{!v.showConfirmDialog}">-->

<!--Modal Box Content Start-->

<div id="divExcel" class="slds-modal\_\_content slds-p-around--medium">

</div>

<!--Modal Box Content End-->

<!--<aura:if isTrue = "{!v.showConfirmDialog}">-->

<!--Modal Box Button Start-->

<footer class="slds-modal\_\_footer">

<lightning:button variant="brand" name='Yes' label='Guardar' onclick='{!c.readCSV}'/>

<lightning:button name='No' label='Cancelar' onclick='{!c.closeModel}'/>

</footer>

<!--Modal Box Button End-->

</div>

</div>

<div class="slds-backdrop slds-backdrop--open"></div>

</aura:if>

</aura:if>

</div>

<!--Confirmation Pop-Up section Start End-->

</aura:component>

**Controller:**

({

// Showing the values in Popup while uploading files in both .csv and .xlsx format

showexceldata : function (component,event,helper){

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

const headers2 = new Array("Nombredecapacitacion", "Nombredeentrenador", "TipodeEntrenamiento", "Fecha", "Esuntemarelacionadoalanutricion" , "Numerodeidentificacion");

console.log('headers2>>'+ headers2 );

var textdata;

var reader = new FileReader();

var xlsxreader = new FileReader();

reader.onload = function (e) {

// alert('onload reader');

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var table = document.createElement("table");

//alert('table>>'+ table);

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

// alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

//alert("Please Upload Valid File ");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos " ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 6){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 0; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

var row = table.insertRow(-1);

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

var cell = row.insertCell(-1);

cell.innerHTML = data[j];

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

//arr.push(obj);

//alert('arr>>' + arr);

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

reader.onerror = function (evt) {

console.log("error reading file");

}

xlsxreader.onload = function (e) {

// alert('onload xlsreader');

const headers2 = new Array("Nombredecapacitacion", "Nombredeentrenador", "TipodeEntrenamiento", "Fecha", "Esuntemarelacionadoalanutricion" , "Numerodeidentificacion");

console.log('headers2>>'+ headers2 );

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// alert('bytesLength'+bytesLength);

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

//alert('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var table = document.createElement("table");

// alert('table>>'+ table);

var rows = csvObj.split("\n");

var headers = rows[0].split(',');

console.log('excel headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 6){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for (var i = 0; i < rows.length; i++) {

var cells = rows[i].split(",");

if (cells.length > 1) {

var row = table.insertRow(-1);

for (var j = 0; j < cells.length; j++) {

var cell = row.insertCell(-1);

cell.innerHTML = cells[j];

}

}

}

var divExcel = document.getElementById("divExcel");

divExcel.innerHTML = "";

divExcel.appendChild(table);

}

xlsxreader.onerror = function (evt) {

console.log("error reading file");

}

if(file){

component.set("v.closeModal", true);

component.set("v.showcard", true);

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

},

doInit : function(component, event, helper) {

helper.addFarmerRecord(component, event);

var action = component.get("c.getNutritiontopicMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeOfNRTMap = [];

var TypeOfNRTOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

TypeOfNRTMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeOfNRTOptions.push(state);

}

component.set("v.TypeOfNRTMap", TypeOfNRTMap);

component.set("v.options\_TypeOfNRT", TypeOfNRTOptions);

}

});

$A.enqueueAction(action);

},

handleCompanyOnChange : function(component, event, helper) {

var indutry = component.get("v.opp.Age\_\_c");

},

handleCompanyOnChange1 : function(component, event, helper) {

var indutry = component.get("v.opp.Gender\_\_c");

},

handleCompanyOnChange2 : function(component, event, helper) {

var indutry = component.get("v.opp.Type\_of\_Training\_\_c");

},

doTypeOfGender : function(component, event, helper) {

var action = component.get("c.getGendermapMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeOfGenderMap = [];

var TypeOfGenderOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

TypeOfGenderMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeOfGenderOptions.push(state);

}

component.set("v.TypeOfGenderMap", TypeOfGenderMap);

component.set("v.options\_TypeOfGender", TypeOfGenderOptions);

}

});

$A.enqueueAction(action);

},

doAgetype : function(component, event, helper) {

var action = component.get("c.getAgemapMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeOfAgeMap = [];

var TypeOfAgeOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

TypeOfAgeMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeOfAgeOptions.push(state);

}

component.set("v.TypeOfAgeMap", TypeOfAgeMap);

component.set("v.options\_TypeOfAge", TypeOfAgeOptions);

}

});

$A.enqueueAction(action);

},

doTypeOfTraining : function(component, event, helper) {

var action = component.get("c.getTypeOfTrainingmapMethod");

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

var result = response.getReturnValue();

var TypeOfTOTMap = [];

var TypeOfTOTOptions = [];

for(var i=0; i<result.length; i++) {

}

for(var key in result){

TypeOfTOTMap.push({key: key, value: result[key]});

var state = {

"label": result[key],

"value": key

};

TypeOfTOTOptions.push(state);

}

component.set("v.TypeOfTOTMap", TypeOfTOTMap);

component.set("v.options\_TypeOfTOT", TypeOfTOTOptions);

}

});

$A.enqueueAction(action);

},

//Select all records

handleSelectAll: function(component, event, helper) {

var getID = component.get("v.FarmerList");

var checkvalue = component.find("selectAll").get("v.value");//got the index of selected record

var checkFarmer = component.find("checkFarmer");

if(checkvalue == true){

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",true);

}

}

else{

for(var i=0; i<checkFarmer.length; i++){

checkFarmer[i].set("v.value",false);

}

}

},

addRow: function(component, event, helper) {

helper.addFarmerRecord(component, event);

},

removeRow: function(component, event, helper) {

//Get the Farmer list

var FarmerList = component.get("v.FarmerList");

//Get the target object

var selectedItem = event.currentTarget;

//Get the selected item index

var index = selectedItem.dataset.record;

FarmerList.splice(index, 1);

component.set("v.FarmerList", FarmerList);

},

save: function(component, event, helper) {

var FarmerList = component.get("v.FarmerList");

if (helper.validateFarmerRecords(component, event)) {

//helper.saveFarmerList(component, event);

var NOTRValue = component.get("v.selectedContactId");

var NutritionValue = component.get("v.IsNutrition");

var DateValue = component.get("v.TRDate");

// checking validation for required fields trainerDate,Nutrition related topic,trainerName

if(typeof NOTRValue == 'undefined' || typeof NutritionValue == 'undefined' ||

typeof DateValue == 'undefined' ){

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

else{

// Adding Type of Trainer and Trainer all values adding to FarmerList

var FarmerList = component.get("v.FarmerList");

for (var i = 0; i < FarmerList.length; i++) {

FarmerList[i].Name\_of\_Trainer\_\_c = NOTRValue ;

FarmerList[i].Es\_este\_un\_tema\_relacionado\_con\_la\_nutr\_\_c = NutritionValue;

FarmerList[i].Training\_date\_\_c = DateValue;

}

////Call Apex class and pass Farmer list parameters

var action = component.get("c.saveExtensionForm");

action.setParams({

"Farmlist": component.get("v.FarmerList"),

});

action.setCallback(this, function(response) {

var state = response.getState();

if (state === "SUCCESS") {

component.set("v.FarmerList", []);

//alert('Registros guardados ');//eng translation for alert:Records Saved Successfully

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

$A.get('e.force:refreshView').fire();

}

});

$A.enqueueAction(action);

}

}

},

//Method to show confirmation pop-up

handleConfirmDialog : function(component, event, helper) {

component.set('v.showConfirmDialog', true);

},

//If user clicked Yes on confirmation popup

handleConfirmDialogYes : function(component, event, helper) {

console.log('Yes');

component.set('v.showConfirmDialog', false);

helper.removeRow(component, event);

},

//If user clicked No on confirmation popup

handleConfirmDialogNo : function(component, event, helper) {

console.log('No');

component.set('v.showConfirmDialog', false);

},

captureNutri :function(component,event){

var StateValue = event.getParam("value");

//alert(StateValue);

component.set("v.IsNutrition", StateValue);

},

captureNOTR :function(component,event){

var NOTGValue = event.getParam("value");

//alert(NOTGValue);

component.set("v.NameOTR", NOTGValue);

},

captureDate :function(component,event){

var TGDateValue = event.getParam("value");

//alert(TGDateValue);

component.set("v.TRDate", TGDateValue);

},

closeModel: function(component, event, helper) {

component.set("v.closeModal", false);

//location.reload();

},

// Validating and Saving the records while uploading both .csv and .xlsx format

readCSV: function(component, event, helper) {

const headers2 = new Array("Nombredecapacitacion", "Nombredeentrenador", "TipodeEntrenamiento", "Fecha", "Esuntemarelacionadoalanutricion" , "Numerodeidentificacion");

console.log('headers2>>'+ headers2 );

var fileInput = component.find("file").getElement();

var file = fileInput.files[0];

var textdata;

var xlsxreader = new FileReader();

var reader = new FileReader();

var infolst = [];

var infolstmain = [];

reader.onload = function() {

// alert(reader);

var text = reader.result; /\*Get the data stored in file\*/

//alert(text);

console.log(text);

textdata = text;

var rows = textdata.split('\n'); /\*Spilt based on new line to get each row\*/

//alert('rows>>'+ rows);

var arr=[];

rows.pop();

//alert('rows>>' + rows);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

var headers = rows[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 6){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < rows.length; k++) {

var data = rows[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

arr.push(obj);

// alert('arr>>' + arr);

}

var data1 = arr;

for( var k=0;k<arr.length;k++){

console.log(data1);

if(data1[k].Nombredecapacitacion=="" ||

data1[k].Nombredeentrenador==""||

data1[k].TipodeEntrenamiento=="" ||

data1[k].Fecha=="" ||

data1[k].Esuntemarelacionadoalanutricion=="" ||

data1[k].Numerodeidentificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var jsonstr = JSON.stringify(arr);

console.log('jsonstr' +jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

var Nombredecapacitacion;

var Nombredeentrenador;

var TipodeEntrenamiento;

var Fecha;

var Esuntemarelacionadoalanutricion;

var Numerodeidentificacion;

xlsxreader.onload = function(e){

var binary = "";

var bytes = new Uint8Array(e.target.result); // data read as bytes

var bytesLength = bytes.byteLength;

// converting to a binary

for (var i = 0; i < bytesLength; i++) {

binary += String.fromCharCode(bytes[i]);

}

var workbook = XLSX.read(binary, { type: 'binary',cellDates: true});

console.log('workbook:'+workbook);

var sheet\_name\_list = workbook.SheetNames;

console.log('sheet\_name\_list:'+sheet\_name\_list);

var csvObj = XLSX.utils.sheet\_to\_csv(workbook.Sheets[sheet\_name\_list[0]]);

console.log('@csv file:'+csvObj.length);

var arr = csvObj.split('\n');;

//var arr = [];

var rows =[]; /\*Spilt based on new line to get each row\*/

//alert(rows);

arr.pop();

//alert('arr>>'+ arr);

//var jsonObj = [];

var headers = arr[0].split(',');

//alert('headers>>' + headers);

for(var i=0; i < headers2.length ; i++){

console.log(headers2[i]);

console.log(headers[i]);

if(headers2[i].trim() != headers[i].trim() ){

// alert("Please Upload Valid File");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "por favor suba archivos válidos ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

//console.log('Values Mismatched');

}

}

if(headers.length!= 6){ // here we validate extra field are in csv or xlsx file

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "elimine la columna adicional" ,

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

for(var k = 1; k < arr.length; k++) {

var data = arr[k].split(',');

// alert('data>>'+data);

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

rows.push(obj);

// alert('Rows>>' + rows);

}

var data1 = rows;

for( var k=0;k<rows.length;k++){

console.log(data1);

if(data1[k].Nombredecapacitacion=="" ||

data1[k].Nombredeentrenador==""||

data1[k].TipodeEntrenamiento=="" ||

data1[k].Fecha=="" ||

data1[k].Esuntemarelacionadoalanutricion=="" ||

data1[k].Numerodeidentificacion==""){

// alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

var Nombredecapacitacion;

var Nombredeentrenador;

var TipodeEntrenamiento;

var Fecha;

var Esuntemarelacionadoalanutricion;

var Numerodeidentificacion;

var infolst = [];

var jsonstr = JSON.stringify(rows);

console.log('jsonstr' +jsonstr);

// alert('before create contact rows'+ jsonstr);

var readerxlx = jsonstr;

$A.getCallback(function() {

if (component.isValid()) {

helper.CreateContact(component, readerxlx);

}

})();

};

// if (filename[0] !== undefined && filename[0] !== null && filename[0] !== '') {

if (file !== undefined && file !== null && file !== '') {

var name= file.name;

var filetype = name.substring(name.lastIndexOf("."),name.length);

if(filetype === '.xlsx' || filetype === '.xls' ){

//console.log('Type'+filetype);

xlsxreader.readAsArrayBuffer(file);

}else{

//console.log('Type'+'.csv');

reader.readAsText(file);

}

}

component.set("v.closeModal", false);

component.set("v.showcard", false);

},

})

**Helper:**

({

CSV2JSON: function (component,csv) {

// console.log('Incoming csv = ' + csv);

//var array = [];

var arr = [];

arr = csv.split('\n');

//console.log('Array = '+array);

// console.log('arr = '+arr);

arr.pop();

var jsonObj = [];

var headers = arr[0].split(',');

for(var i = 1; i < arr.length; i++) {

var data = arr[i].split(',');

var obj = {};

for(var j = 0; j < data.length; j++) {

obj[headers[j].trim()] = data[j].trim();

//console.log('obj headers = ' + obj[headers[j].trim()]);

}

jsonObj.push(obj);

}

var data1 = jsonObj;

for( var k=0;k<jsonObj.length;k++){

console.log(data1);

if(data1[k].Nombredecapacitacion=="" ||

data1[k].Nombredeentrenador==""||

data1[k].TipodeEntrenamiento=="" ||

data1[k].fecha=="" ||

data1[k].Esuntemarelacionadoalanutricion=="" ||

data1[k].Identificacion==""){

//alert('Las celdas obligatorias no pueden estar en blanco en la fila');

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila " + (k+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

return null;

}

}

/\* isValid = false;

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila ",

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

// return isValid;

}

}

return isValid;\*/

var json = JSON.stringify(data1);

console.log('json = '+ json);

return json;

},

CreateContact : function (component,jsonstr){

// console.log('jsonstr' + jsonstr);

var action = component.get('c.insertData');

// alert('Server Action' + action);

action.setParams({

strfromle : jsonstr

});

action.setCallback(this, function(response) {

var state = response.getState();

//alert(state);

if (state === "SUCCESS") {

var result=response.getReturnValue();

// alert('result callback>>'+ result);

// alert("Records Inserted Succesfully");

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Éxito!",

"message": "Registros guardados.",

duration: '3000',

key: "info\_alt",

type: "success",

mode: "pester"

});

toastEvent.fire();

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

//console.log("Error message: " + errors[0].message);

}

} else {

//console.log("Unknown error");

//alert('Unknown');

}

}

});

$A.enqueueAction(action);

},

addFarmerRecord: function(component, event) {

//get the Farmer List from component

var FarmerList = component.get("v.FarmerList");

//Add New Farmer Record

FarmerList.push({

'sobjectType': 'Farmers\_\_c',

'Contact\_Name\_\_c':'',

//'Edad\_\_c':'',

//'Genero\_\_c':'',

'Es\_este\_un\_tema\_relacionado\_con\_la\_nutr\_\_c': '',

'Trainer\_name\_\_c':'',

'Training\_date\_\_c':'',

'Name\_of\_training\_\_c':'',

'Tipo\_de\_entrenamiento\_extension\_\_c' : '',

'Contact\_Name\_\_c' : '',

});

component.set("v.FarmerList", FarmerList);

},

//Process the selected Farmers records to remove

removeRow: function(component, event) {

var selectedFarmers = component.get("v.FarmerList");

//temporary array to hold non-selected reocrds

var finalFarmers = [];

var checkvalue = component.find("checkFarmer");

//if single records is selected else its a array

if(!Array.isArray(checkvalue)){

if (checkvalue.get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue.get("v.text")])

}

}else{

for (var i = 0; i < checkvalue.length; i++) {

if (checkvalue[i].get("v.value") == false) {

finalFarmers.push(selectedFarmers[checkvalue[i].get("v.text")]);

}

}

}

console.log('finalFarmers-' + finalFarmers);

//Now copy the non selected rows to primary array

component.set("v.FarmerList", finalFarmers);

},

validateFarmerRecords: function(component, event) {

//Validate all account records

var isValid = true;

var FarmerList = component.get("v.FarmerList");

//alert(FarmerList[i].State\_\_c);

//alert( FarmerList[i].Applying\_new\_technique\_\_c);

for (var i = 0; i < FarmerList.length; i++) {

if (FarmerList[i].Contact\_Name\_\_c == '' ||

FarmerList[i].Edad\_\_c == '' ||

FarmerList[i].Genero\_\_c == '' ||

FarmerList[i].Name\_of\_training\_\_c == '' ||

FarmerList[i].Tipo\_de\_entrenamiento\_extension\_\_c== '' )

{

isValid = false;

//alert('Las celdas obligatorias no pueden estar en blanco'+(i+1));// eng translate for alert: Mandatory feilds cannot be blank

var toastEvent = $A.get("e.force:showToast");

toastEvent.setParams({

"title": "Error!",

"message": "Las celdas obligatorias no pueden estar en blanco en la fila" + (i+1),

duration: '3000',

key: "info\_alt",

type: "error",

mode: "pester"

});

toastEvent.fire();

}

}

return isValid;

},

})

**Css:**

.THIS .date .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .date .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"Utilice el siguiente formato: DD/MM/AAAA";

}

.THIS .date .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .message .slds-form-element\_\_help {

visibility: hidden;

position: relative;

left: 15px;

}

.THIS .message .slds-form-element\_\_help:after {

visibility: visible;

position: absolute;

left: 10px;

top: 0;

content:"complete este campo";

}

.THIS .message .slds-form-element\_\_help:before {

position: absolute;

visibility: visible;

left: -5px;

top: 0;

font-style: normal;

font-size: 18px;

content: "\25b8";

top: -3.5px;

}

.THIS .slds-container--x--large{

box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-moz-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-webkit-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

-o-box-shadow: 0 0 10px rgba(0,0,0,0.6) !important;

}

.THIS input[type="file"] {

display: none;

}

.THIS.custom-file-upload {

color: #333;

margin: 1px;

padding: 2px 3px;

border: 1px solid #b5b5b5;

border-bottom-color: #7f7f7f;

background: #e8e8e9;

font-weight: bold;

font-size: .9em;

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

padding: 4px 3px;

cursor: pointer;

}

**ApexController:**

public class Extension\_Agent\_ParticipantController {

/\* for import data into sobject and insert data from csv file \*/

@AuraEnabled

public static string insertData(String strfromle){

String returnresponse ='';

List<Farmers\_\_c> ListtoInsert = new List<Farmers\_\_c>();

system.debug('strfromle = ' + strfromle);

list< fieldWrapper> datalist = (list<fieldWrapper>)JSON.deserialize(strfromle, List<fieldWrapper>.class);

system.debug('datalist = ' + datalist);

//system.debug('datalistSize = ' + datalist.size());

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Extension Agent Participant Training Form' limit 1];

for(fieldWrapper wrapper: datalist){

Farmers\_\_c far1 =new Farmers\_\_c();

system.debug('wrapper'+wrapper);

far1.RecordTypeId =RT.Id;

far1.Name\_of\_training\_\_c = wrapper.Nombredecapacitacion;

far1.ExtenAgent\_\_c = wrapper.Nombredeentrenador;

far1.Tipo\_de\_entrenamiento\_extension\_\_c = wrapper.TipodeEntrenamiento;

far1.Training\_date\_\_c= wrapper.Fecha;

far1.Es\_este\_un\_tema\_relacionado\_con\_la\_nutr\_\_c = wrapper.Esuntemarelacionadoalanutricion;

if(wrapper.Numerodeidentificacion!= null){

far1.Code\_\_c = wrapper.Numerodeidentificacion;

}

ListtoInsert.add(far1);

}

if(ListtoInsert.size() > 0){

try {

insert ListtoInsert;

returnresponse = 'SUCCESS';

}

catch(Exception ex){

returnresponse = 'ERROR';

}

}

return returnresponse;

}

public class fieldWrapper{

public String Nombredecapacitacion;

public String Nombredeentrenador;

public String TipodeEntrenamiento;

public String Fecha;

public String Esuntemarelacionadoalanutricion;

public String Numerodeidentificacion;

}

@AuraEnabled

public static void saveExtensionForm(list<Farmers\_\_c> Farmlist){

system.debug('Farmlist>>'+Farmlist);

list<Farmers\_\_c> finalFarmlist = new list<Farmers\_\_c> ();

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Extension Agent Participant Training Form' limit 1];

for(Farmers\_\_c far : Farmlist){

far.Name = 'test';

far.RecordTypeId = RT.Id;

finalFarmlist.add(far);

}

system.debug('finalFarmlist>>'+finalFarmlist);

Insert finalFarmlist;

}

@AuraEnabled

public static Map<String, String> getNutritiontopicMethod(){

Map<String, String> optionsOfNRTmap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Es\_este\_un\_tema\_relacionado\_con\_la\_nutr\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsOfNRTmap.put(ple.getValue(), ple.getLabel());

}

return optionsOfNRTmap;

}

@AuraEnabled

public static Map<String, String> getTypeOfTrainingmapMethod(){

Map<String, String> optionsOfTypeTrgmap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Tipo\_de\_entrenamiento\_extension\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsOfTypeTrgmap.put(ple.getValue(), ple.getLabel());

}

return optionsOfTypeTrgmap;

}

@AuraEnabled

public static Map<String, String> getAgemapMethod(){

Map<String, String> optionsAgemap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Edad\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsAgemap.put(ple.getValue(), ple.getLabel());

}

return optionsAgemap;

}

@AuraEnabled

public static Map<String, String> getGendermapMethod(){

Map<String, String> optionsOfGendermap = new Map<String, String>();

Schema.DescribeFieldResult fieldResult = Farmers\_\_c.Genero\_\_c.getDescribe();

List<Schema.PicklistEntry> pleList = fieldResult.getPicklistValues();

for (Schema.PicklistEntry ple : pleList) {

optionsOfGendermap.put(ple.getValue(), ple.getLabel());

}

return optionsOfGendermap;

}

}

**TestClass:**

@isTest

public class ExtensionAgentParticipantControllerTest {

@istest

static void ExtensionAgentTest(){

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Extension Agent Participant Training Form' limit 1];

system.debug('RT@@'+RT);

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far1=new Farmers\_\_c();

far1.Name='test';

far1.Es\_este\_un\_tema\_relacionado\_con\_la\_nutr\_\_c='No';

far1.Trainer\_name\_\_c ='test123';

far1.Training\_date\_\_c='12/02/2022';

far1.Name\_of\_training\_\_c='Test';

far1.Tipo\_de\_entrenamiento\_extension\_\_c ='Aprendizaje de adultos';

far1.Contact\_Name\_\_c= con.id;

far1.RecordTypeId = RT.Id;

insert finalFarmlist1;

Extension\_Agent\_ParticipantController.saveExtensionForm(finalFarmlist1);

Map<String, String> getNutri = Extension\_Agent\_ParticipantController.getNutritiontopicMethod();

system.assertNotEquals(Null, getNutri);

Map<String, String> getTraining = Extension\_Agent\_ParticipantController.getTypeOfTrainingmapMethod();

system.assertNotEquals(Null, getTraining);

Map<String, String> getAge = Extension\_Agent\_ParticipantController.getAgemapMethod();

system.assertNotEquals(Null, getAge);

Map<String, String> getGender = Extension\_Agent\_ParticipantController.getGendermapMethod();

system.assertNotEquals(Null, getGender);

}

@istest

static void ExtensionAgentCSVTest(){

String returnresponse ='';

//Contact Creation

contact con = new contact();

con.lastname='test123 name';

con.email='test123@gmail.com';

insert con;

RecordType RT = [SELECT Id, Name FROM RecordType where Name = 'Extension Agent Participant Training Form' limit 1];

system.debug('RT@@'+RT);

list<Farmers\_\_c> finalFarmlist1 = new list<Farmers\_\_c> ();

list<Farmers\_\_c> finalFarmlist2 = new list<Farmers\_\_c> ();

Farmers\_\_c far2=new Farmers\_\_c();

far2.Name='test';

far2.Es\_este\_un\_tema\_relacionado\_con\_la\_nutr\_\_c='No';

far2.Trainer\_name\_\_c ='test123';

far2.Training\_date\_\_c='12/02/2022';

far2.Name\_of\_training\_\_c='Test';

far2.Tipo\_de\_entrenamiento\_extension\_\_c ='Aprendizaje de adultos';

far2.Contact\_Name\_\_c= con.id;

far2.RecordTypeId = RT.Id;

finalFarmlist1.add(far2);

insert finalFarmlist1;

// Extension\_Agent\_ParticipantController.saveExtensionForm(finalFarmlist1);

Extension\_Agent\_ParticipantController.insertData(JSON.serialize(finalFarmlist1));

Map<String, String> getNutri = Extension\_Agent\_ParticipantController.getNutritiontopicMethod();

system.assertNotEquals(Null, getNutri);

Map<String, String> getTraining = Extension\_Agent\_ParticipantController.getTypeOfTrainingmapMethod();

system.assertNotEquals(Null, getTraining);

Map<String, String> getAge = Extension\_Agent\_ParticipantController.getAgemapMethod();

system.assertNotEquals(Null, getAge);

Map<String, String> getGender = Extension\_Agent\_ParticipantController.getGendermapMethod();

system.assertNotEquals(Null, getGender);

}

}