/\*\*

\* ✅ SOLUTION DEBUG : Affichage des erreurs serveur dans le modal d'alerte

\* À ajouter/modifier dans le JavaScript de la page de création d'organisation

\*/

// ✅ FONCTION AMÉLIORÉE : Afficher les erreurs détaillées du serveur

function showErrorModal(title, message, serverResponse = null, isDebug = true) {

// Construire le message d'erreur détaillé

let fullMessage = message;

if (isDebug && serverResponse) {

fullMessage += '\n\n' + '='.repeat(50);

fullMessage += '\n🔍 DEBUG - RÉPONSE SERVEUR DÉTAILLÉE :';

fullMessage += '\n' + '='.repeat(50);

// Afficher les détails de la réponse

if (typeof serverResponse === 'object') {

try {

// Si c'est un objet, l'afficher en JSON formaté

fullMessage += '\n📄 Contenu de la réponse :\n';

fullMessage += JSON.stringify(serverResponse, null, 2);

// Afficher les erreurs spécifiques si disponibles

if (serverResponse.errors) {

fullMessage += '\n\n🚨 Erreurs de validation :\n';

Object.keys(serverResponse.errors).forEach(field => {

fullMessage += `• ${field}: ${serverResponse.errors[field].join(', ')}\n`;

});

}

// Afficher les infos de debug si disponibles

if (serverResponse.debug) {

fullMessage += '\n\n🐛 Informations de debug :\n';

fullMessage += `Fichier: ${serverResponse.debug.file}\n`;

fullMessage += `Ligne: ${serverResponse.debug.line}\n`;

if (serverResponse.debug.json\_error) {

fullMessage += `Erreur JSON: ${serverResponse.debug.json\_error}\n`;

}

}

} catch (e) {

fullMessage += '\n📄 Réponse brute :\n' + String(serverResponse);

}

} else {

fullMessage += '\n📄 Réponse brute :\n' + String(serverResponse);

}

fullMessage += '\n' + '='.repeat(50);

}

// Créer et afficher le modal d'erreur amélioré

const modal = `

<div class="modal fade" id="errorDebugModal" tabindex="-1" role="dialog" style="z-index: 9999;">

<div class="modal-dialog modal-lg" role="document">

<div class="modal-content border-danger">

<div class="modal-header bg-danger text-white">

<h5 class="modal-title">

<i class="fas fa-exclamation-triangle me-2"></i>

${title}

</h5>

<button type="button" class="btn-close btn-close-white" data-bs-dismiss="modal"></button>

</div>

<div class="modal-body">

<div class="alert alert-danger">

<h6 class="alert-heading">

<i class="fas fa-bug me-2"></i>

Erreur de Soumission - Mode Debug

</h6>

<pre style="white-space: pre-wrap; font-family: monospace; font-size: 12px; max-height: 400px; overflow-y: auto; background: #f8f9fa; padding: 15px; border-radius: 5px;">${fullMessage}</pre>

</div>

<div class="mt-3">

<button type="button" class="btn btn-outline-primary btn-sm" onclick="copyDebugInfo()">

<i class="fas fa-copy me-1"></i>

Copier les détails

</button>

<button type="button" class="btn btn-outline-secondary btn-sm" onclick="console.log('Debug Info:', ${JSON.stringify(serverResponse || {})})">

<i class="fas fa-terminal me-1"></i>

Afficher dans la console

</button>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Fermer</button>

<button type="button" class="btn btn-primary" onclick="location.reload()">

<i class="fas fa-refresh me-1"></i>

Recharger la page

</button>

</div>

</div>

</div>

</div>

`;

// Supprimer le modal existant s'il y en a un

const existingModal = document.getElementById('errorDebugModal');

if (existingModal) {

existingModal.remove();

}

// Ajouter le nouveau modal

document.body.insertAdjacentHTML('beforeend', modal);

// Afficher le modal

const modalElement = new bootstrap.Modal(document.getElementById('errorDebugModal'));

modalElement.show();

// Stocker les infos de debug globalement pour la fonction de copie

window.debugInfo = fullMessage;

}

// ✅ FONCTION : Copier les informations de debug

function copyDebugInfo() {

if (window.debugInfo) {

navigator.clipboard.writeText(window.debugInfo).then(() => {

alert('Informations de debug copiées dans le presse-papier !');

}).catch(() => {

// Fallback pour les navigateurs plus anciens

const textarea = document.createElement('textarea');

textarea.value = window.debugInfo;

document.body.appendChild(textarea);

textarea.select();

document.execCommand('copy');

document.body.removeChild(textarea);

alert('Informations de debug copiées !');

});

}

}

// ✅ MODIFICATION DE LA FONCTION DE SOUMISSION AJAX

function submitOrganisationForm() {

// ... code existant pour collecter les données ...

// Envoi AJAX avec gestion d'erreur améliorée

$.ajax({

url: '/operator/organisations',

method: 'POST',

data: formData,

processData: false,

contentType: false,

headers: {

'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')

},

beforeSend: function() {

// Afficher le loader

showLoadingModal();

},

success: function(response) {

console.log('✅ Succès de soumission:', response);

if (response.success) {

// Redirection vers la page de confirmation

if (response.data && response.data.redirect\_url) {

console.log('🔄 Redirection vers:', response.data.redirect\_url);

window.location.href = response.data.redirect\_url;

} else {

// Fallback si pas d'URL de redirection

showErrorModal(

'Succès mais redirection manquante',

'L\'organisation a été créée avec succès mais l\'URL de redirection est manquante.',

response,

true

);

}

} else {

showErrorModal(

'Erreur de traitement',

response.message || 'Une erreur est survenue lors du traitement.',

response,

true

);

}

},

error: function(xhr, status, error) {

console.error('❌ Erreur AJAX:', {xhr, status, error});

let errorTitle = 'Erreur de Soumission';

let errorMessage = 'Une erreur est survenue lors de la soumission du formulaire.';

let serverResponse = null;

try {

// Tenter de parser la réponse JSON du serveur

serverResponse = JSON.parse(xhr.responseText);

if (serverResponse.message) {

errorMessage = serverResponse.message;

}

// Ajouter des détails sur le statut HTTP

errorMessage += `\n\nStatut HTTP: ${xhr.status} (${xhr.statusText})`;

} catch (e) {

// Si ce n'est pas du JSON, utiliser le texte brut

serverResponse = {

status: xhr.status,

statusText: xhr.statusText,

responseText: xhr.responseText,

readyState: xhr.readyState

};

errorMessage += `\n\nErreur: ${error}`;

errorMessage += `\nStatut: ${status}`;

errorMessage += `\nCode HTTP: ${xhr.status}`;

}

// Afficher le modal d'erreur avec tous les détails

showErrorModal(errorTitle, errorMessage, serverResponse, true);

},

complete: function() {

// Cacher le loader

hideLoadingModal();

}

});

}

// ✅ FONCTION ALTERNATIVE : Pour les soumissions avec fetch()

function submitWithFetch() {

fetch('/operator/organisations', {

method: 'POST',

body: formData,

headers: {

'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').getAttribute('content')

}

})

.then(async response => {

const contentType = response.headers.get('content-type');

let responseData = null;

try {

if (contentType && contentType.includes('application/json')) {

responseData = await response.json();

} else {

responseData = await response.text();

}

} catch (e) {

responseData = 'Impossible de lire la réponse du serveur';

}

if (!response.ok) {

throw {

status: response.status,

statusText: response.statusText,

data: responseData

};

}

return responseData;

})

.then(data => {

console.log('✅ Succès:', data);

if (data.success && data.data && data.data.redirect\_url) {

console.log('🔄 Redirection vers:', data.data.redirect\_url);

window.location.href = data.data.redirect\_url;

} else {

showErrorModal(

'Réponse inattendue',

'La soumission semble avoir réussi mais la réponse est inattendue.',

data,

true

);

}

})

.catch(error => {

console.error('❌ Erreur Fetch:', error);

showErrorModal(

'Erreur de Soumission',

'Une erreur est survenue lors de la soumission.',

error,

true

);

});

}

// ✅ LOGS DE DEBUG AMÉLIORÉS

console.log('🔧 Module de debug des erreurs de soumission chargé');

console.log('📝 Fonctions disponibles: showErrorModal(), copyDebugInfo(), submitOrganisationForm()');

// ✅ ACTIVATION DU MODE DEBUG GLOBAL

window.DEBUG\_MODE = true;

window.showErrorModal = showErrorModal;