Voici le document d'état du projet, mis à jour avec tous les échanges de notre session. Il est prêt à être sauvegardé pour notre prochaine collaboration.

Markdown

---

Projet: "Usine à Tests"

Date: "2508220238"

Version: "19.4"

---

### 1. Résumé des Échanges du Jour

La session a été exceptionnellement dense et productive, axée sur la finalisation de la robustesse du code existant et le développement complet de la fonctionnalité d'\*\*envoi différé des e-mails\*\*.

Le parcours a été le suivant :

1. \*\*Finalisation du Débogage (E-mails non reçus)\*\* : Nous avons commencé par résoudre le problème persistant des e-mails non reçus par le formateur. L'ajout de logs (`Logger.log`) nous a permis de remonter la chaîne des erreurs :

\* Une clé manquante (`PREFIXE\_COPIE\_EMAIL`) dans l'onglet `traductions`.

\* Une erreur due à un espace dans un en-tête de colonne, nous amenant à renforcer la fonction `loadTraductions` avec `.trim()`.

\* Un problème de pièce jointe manquante, qui s'est avéré ne pas être une erreur d'API aléatoire, mais une erreur de configuration de donnée.

2. \*\*Le Moment "EUREKA"\*\* : Après une investigation poussée, vous avez découvert la cause racine du problème de pièce jointe : un \*\*caractère de saut de ligne invisible\*\* dans la cellule de l'ID du document dans l'onglet `sys\_Composition\_Emails`. Cela a prouvé que le problème n'était pas l'ID lui-même, mais la manière dont il était lu.

3. \*\*Développement de l'Envoi Différé\*\* : Nous avons ensuite implémenté de bout en bout la fonctionnalité de temporisation :

\* Analyse de la source des options de délai (`sys\_Options\_Parametres`).

\* Création d'un nouveau script `GestionTriggers.gs` pour isoler la logique de programmation.

\* Modification de `TraitementReponses.gs` pour lire la configuration de délai et appeler le nouveau gestionnaire de déclencheurs.

\* Test complet du mécanisme (programmation, exécution, envoi et auto-nettoyage du trigger).

4. \*\*Améliorations de Robustesse\*\* :

\* Le script `TraitementReponses.gs` a été modifié pour que le retraitement manuel force toujours un envoi immédiat, ignorant la configuration de délai.

\* Il a également été renforcé pour nettoyer automatiquement les ID de pièces jointes (`.trim()`), le rendant insensible aux erreurs de saisie comme les sauts de ligne.

5. \*\*Clarification de l'Architecture\*\* : Nous avons clarifié que le "Moteur Universel" actuel utilise exclusivement `sys\_Composition\_Emails` pour les pièces jointes, rendant l'onglet `sys\_PiecesJointes` obsolète pour ce moteur.

---

### 2. État Actuel du Projet

\* \*\*Tâches réalisées\*\* :

\* Développement complet, test et validation de la fonctionnalité d'\*\*envoi différé des résultats\*\*.

\* Correction finale et définitive du bug des \*\*pièces jointes manquantes\*\*.

\* Amélioration significative de la \*\*robustesse du code\*\* face aux erreurs de saisie de données (espaces, sauts de ligne).

\* Mise en place d'une \*\*version stable et complète (v19.4)\*\* de l'ensemble des scripts du template.

\* \*\*Tâches en cours\*\* :

\* Aucune. La fonctionnalité "Envoi Différé" est considérée comme terminée.

\* \*\*Prochaines étapes planifiées\*\* :

\* Nettoyage des données de test (suppression de l'option `1min` dans la configuration).

\* Prise de décision sur la prochaine fonctionnalité à développer.

\* Réflexion sur le nettoyage du code "mort" (la fonction `findAttachments` et l'onglet `sys\_PiecesJointes`).

---

### 3. Contexte Historique du Projet

Cette session s'inscrit dans la continuité de la phase d'amélioration de l'expérience utilisateur et de la communication. [cite\_start]Après avoir stabilisé la brique "E-mail de confirmation" lors de la session précédente[cite: 435], nous avons aujourd'hui ajouté la brique "Envoi Différé", une fonctionnalité métier à forte valeur ajoutée. Cette session a été marquée par un débogage en profondeur qui a non seulement résolu des problèmes visibles, mais a aussi permis de renforcer considérablement la fiabilité du code face à des erreurs de configuration invisibles.

---

### 4. Orientations Stratégiques

La session a solidifié plusieurs principes d'architecture clés :

\* \*\*Robustesse contre les Données\*\* : Les corrections multiples basées sur `.trim()` renforcent le principe de ne pas faire confiance aux données d'entrée. Le script doit être capable de se "défendre" contre des erreurs de saisie humaines.

\* \*\*Modularité\*\* : La création du fichier `GestionTriggers.gs` pour une fonctionnalité spécifique (la programmation) est une application directe de la séparation des préoccupations, rendant le code plus lisible et plus facile à maintenir.

\* \*\*Investigation par la Donnée\*\* : Notre démarche a prouvé que face à un bug persistant, il est essentiel de questionner les données (`sys\_Composition\_Emails`) autant que le code.

---

### 5. Architecture Technique & Ressources Clés

#### 5.1. État des Scripts et Fichiers

[cite\_start]Voici l'ensemble des scripts du projet "Template" dans leur version finale et stabilisée. [cite: 102]

\*\*Fichier 1 : `TraitementReponses.gs` (Version 19.4)\*\*

```javascript

// =================================================================================

// == FICHIER : TraitementReponses.gs

// == VERSION : 19.4 (Nettoyage automatique des ID de pièces jointes)

// == RÔLE : Gère la logique de traitement des réponses et aiguille vers le bon moteur.

// =================================================================================

/\*\*

\* Nettoie une chaîne de caractères pour la rendre utilisable comme nom de variable/placeholder.

\*/

function \_nettoyerEnTete(enTete) {

if (!enTete) return "";

const accents = 'ÀÁÂÃÄÅàáâãäåÒÓÔÕÖØòóôõöøÈÉÊËèéêëÇçÌÍÎÏìíîïÙÚÛÜùúûüÿÑñ';

const sansAccents = 'AAAAAAaaaaaaOOOOOOooooooEEEEeeeeCcIIIIiiiiUUUUuuuuyNn';

return enTete.toString().split('').map((char) => {

const accentIndex = accents.indexOf(char);

return accentIndex !== -1 ? sansAccents[accentIndex] : char;

}).join('')

.replace(/[^a-zA-Z0-9\_]/g, '\_');

}

/\*\*

\* Crée un objet de réponse standardisé à partir d'un numéro de ligne.

\*/

function \_creerObjetReponse(rowIndex) {

const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheets()[0];

const headers = sheet.getRange(1, 1, 1, sheet.getLastColumn()).getValues()[0];

const rowValues = sheet.getRange(rowIndex, 1, 1, sheet.getLastColumn()).getValues()[0];

const reponse = {};

headers.forEach((header, i) => {

let cle = header;

if (header && !header.includes(':')) {

cle = \_nettoyerEnTete(header);

}

if (cle) {

reponse[cle] = rowValues[i];

}

});

return reponse;

}

/\*\*

\* Point d'entrée principal pour traiter une soumission de formulaire.

\*/

function onFormSubmit(e) {

try {

const rowIndex = e.range.getRow();

traiterLigne(rowIndex, {});

} catch (err) {

Logger.log(`Erreur critique dans onFormSubmit pour la ligne ${e.range.getRow()}: ${err.toString()}\n${err.stack}`);

}

}

/\*\*

\* Envoie un e-mail de confirmation.

\*/

function \_envoyerEmailDeConfirmation(config, reponse, langueCible) {

try {

const nomColonneOverride = `ID\_Gabarit\_Email\_Confirmation\_${langueCible}`;

let idGabaritConfirmation = config[nomColonneOverride];

if (!idGabaritConfirmation || String(idGabaritConfirmation).trim() === '') {

const systemIds = getSystemIds();

const nomCleDefaut = `ID\_GABARIT\_CONFIRMATION\_${langueCible}`;

idGabaritConfirmation = systemIds[nomCleDefaut];

Logger.log(`Utilisation du gabarit de confirmation PAR DÉFAUT pour la langue ${langueCible}.`);

} else {

Logger.log(`Utilisation du gabarit de confirmation SPÉCIFIQUE pour la langue ${langueCible}.`);

}

const emailRepondant = reponse.Votre\_adresse\_e\_mail || reponse.Adresse\_e\_mail || reponse.emailRepondant;

if (!idGabaritConfirmation || String(idGabaritConfirmation).trim() === '' || !emailRepondant) {

Logger.log(`Aucun e-mail de confirmation à envoyer pour la langue ${langueCible}.`);

return;

}

const doc = DocumentApp.openById(idGabaritConfirmation);

let sujet = doc.getName();

const url = "[https://docs.google.com/feeds/download/documents/export/Export?id=](https://docs.google.com/feeds/download/documents/export/Export?id=)" + idGabaritConfirmation + "&exportFormat=html";

const token = ScriptApp.getOAuthToken();

const response = UrlFetchApp.fetch(url, { headers: { 'Authorization': 'Bearer ' + token } });

let corpsHtml = response.getContentText();

for (const key in reponse) {

if (reponse.hasOwnProperty(key)) {

const placeholder = `{{${key}}}`;

const valeur = reponse[key] || '';

const regex = new RegExp(placeholder.replace(/[-\/\\^$\*+?.()|[\]{}]/g, '\\$&'), 'g');

sujet = sujet.replace(regex, valeur);

corpsHtml = corpsHtml.replace(regex, valeur);

}

}

const mailOptions = { to: emailRepondant, subject: sujet, htmlBody: corpsHtml };

if (config.Email\_Alias && config.Email\_Alias.trim() !== '') {

mailOptions.from = config.Email\_Alias;

}

GmailApp.sendEmail(mailOptions.to, mailOptions.subject, "", mailOptions);

Logger.log(`E-mail de confirmation [${langueCible}] envoyé avec succès à ${emailRepondant}.`);

} catch (e) {

Logger.log(`ERREUR lors de l'envoi de l'e-mail de confirmation : ${e.toString()}\n${e.stack}`);

}

}

/\*\*

\* COEUR LOGIQUE : Aiguille le traitement vers le moteur.

\*/

function traiterLigne(rowIndex, optionsSurcharge = {}) {

try {

const config = getTestConfiguration();

const reponse = \_creerObjetReponse(rowIndex);

const langueOrigine = getOriginalLanguage(reponse);

const langueCible = optionsSurcharge.langue || langueOrigine;

if (!optionsSurcharge.isRetraitement) {

\_envoyerEmailDeConfirmation(config, reponse, langueCible);

}

const resultats = calculerResultats(reponse, langueCible, config, langueOrigine);

// Si on est en mode retraitement, on envoie toujours immédiatement.

// Sinon, on respecte la configuration.

if (optionsSurcharge.isRetraitement || config.Repondant\_Quand === 'Immediat') {

if (config.Moteur\_Calcul === 'Universel') {

Logger.log("Moteur UNIVERSEL détecté pour la ligne " + rowIndex + ". Envoi immédiat (retraitement ou configuration).");

assemblerEtEnvoyerEmailUniversel(config, reponse, resultats, langueCible, optionsSurcharge);

} else {

// Moteur legacy...

}

} else {

// Si la valeur n'est pas "Immediat" ET que ce n'est pas un retraitement, on programme.

Logger.log(`Envoi différé détecté. Valeur : "${config.Repondant\_Quand}". Lancement de la programmation.`);

programmerEnvoiResultats(rowIndex, langueCible, config.Repondant\_Quand);

}

} catch (err) {

Logger.log("ERREUR FATALE dans traiterLigne: " + err.toString() + "\n" + err.stack);

}

}

/\*\*

\* Lit la BDD, assemble et envoie l'e-mail de résultats.

\*/

function assemblerEtEnvoyerEmailUniversel(config, reponse, resultats, langueCible, optionsSurcharge = {}) {

const typeTest = config.Type\_Test;

let codeNiveauEmail = config.ID\_Gabarit\_Email\_Repondant.replace('RESULTATS\_', '');

if (optionsSurcharge && optionsSurcharge.niveau && optionsSurcharge.niveau !== '') {

codeNiveauEmail = optionsSurcharge.niveau;

}

const profilFinal = resultats.profilFinal;

const systemIds = getSystemIds();

const bdd = SpreadsheetApp.openById(systemIds.ID\_BDD);

const compoSheet = bdd.getSheetByName("sys\_Composition\_Emails");

const compoData = compoSheet.getDataRange().getValues();

const compoHeaders = compoData.shift();

const idx = {

typeTest: compoHeaders.indexOf('Type\_Test'),

langue: compoHeaders.indexOf('Code\_Langue'),

niveau: compoHeaders.indexOf('Code\_Niveau\_Email'),

profil: compoHeaders.indexOf('Code\_Profil'),

element: compoHeaders.indexOf('Element'),

ordre: compoHeaders.indexOf('Ordre'),

contenu: compoHeaders.indexOf('Contenu / ID\_Document')

};

let briquesDeContenu = compoData.filter(row => {

const typeLigne = (row[idx.typeTest] || '').trim();

const typeCible = (typeTest || '').trim();

const typeMatch = (typeLigne === typeCible || typeLigne === '');

const langMatch = row[idx.langue] === langueCible;

const levelMatch = row[idx.niveau].includes(codeNiveauEmail);

const profilLigne = (row[idx.profil] || '').trim();

const profilCible = (profilFinal || '').trim();

const profileMatch = (profilLigne === profilCible || profilLigne === '');

return typeMatch && langMatch && levelMatch && profileMatch;

});

briquesDeContenu.sort((a, b) => a[idx.ordre] - b[idx.ordre]);

let contenuInfoCopie = null;

const indexInfoCopie = briquesDeContenu.findIndex(brique => (brique[idx.element] || '').trim() === 'Info\_Copie');

if (indexInfoCopie > -1) {

contenuInfoCopie = briquesDeContenu[indexInfoCopie][idx.contenu];

briquesDeContenu.splice(indexInfoCopie, 1);

}

let sujet = `Résultats de votre test ${typeTest}`;

let corpsHtml = "";

const piecesJointesIds = new Set();

for (const brique of briquesDeContenu) {

const elementType = (brique[idx.element] || '').trim();

const contenu = brique[idx.contenu];

switch (elementType) {

case 'Sujet\_Email': sujet = contenu; break;

case 'Introduction': case 'Corps\_Texte': corpsHtml += contenu + "<br>"; break;

case 'Document':

if (contenu && String(contenu).trim()) {

piecesJointesIds.add(String(contenu).trim());

}

break;

case 'Ligne\_Score':

Object.entries(resultats.scoresData).sort((a, b) => b[1] - a[1]).forEach(([code, score]) => {

let ligneScore = contenu.replace(/{{nom\_profil}}/g, resultats.mapCodeToName[code] || code).replace(/{{score}}/g, score);

corpsHtml += ligneScore + "<br>";

});

break;

}

}

const donneesPourEmail = { ...reponse, ...resultats };

for (const key in donneesPourEmail) {

if (donneesPourEmail.hasOwnProperty(key)) {

const placeholder = `{{${key}}}`;

const valeur = donneesPourEmail[key] || '';

const regex = new RegExp(placeholder.replace(/[-\/\\^$\*+?.()|[\]{}]/g, '\\$&'), 'g');

sujet = sujet.replace(regex, valeur);

corpsHtml = corpsHtml.replace(regex, valeur);

if (contenuInfoCopie) {

contenuInfoCopie = contenuInfoCopie.replace(regex, valeur);

}

}

}

const piecesJointes = Array.from(piecesJointesIds).map(id => {

let blob = null;

for (let i = 0; i < 3; i++) { // Tentative de 3 essais

try {

blob = DriveApp.getFileById(id).getBlob();

break; // Succès, on sort de la boucle

} catch (e) {

Logger.log(`Tentative ${i + 1} échouée pour la PJ : ${id}. Erreur: ${e.message}`);

if (i < 2) { // Si ce n'est pas la dernière tentative

Utilities.sleep(1000); // On attend 1 seconde avant de réessayer

} else {

Logger.log(`ERREUR FINALE PJ : Impossible d'attacher le fichier ${id} après 3 tentatives.`);

}

}

}

return blob;

}).filter(Boolean);

const T = loadTraductions(langueCible);

const adressesUniques = new Set();

const emailRepondantPrincipal = reponse.Votre\_adresse\_e\_mail || reponse.Adresse\_e\_mail || reponse.emailRepondant;

const destinatairesSurcharge = optionsSurcharge.destinataires || {};

const useSurcharge = Object.keys(destinatairesSurcharge).length > 0;

if (useSurcharge) {

if (destinatairesSurcharge.repondant && emailRepondantPrincipal) { adressesUniques.add(emailRepondantPrincipal); }

if (destinatairesSurcharge.formateur && destinatairesSurcharge.formateurEmail) { adressesUniques.add(destinatairesSurcharge.formateurEmail); }

if (destinatairesSurcharge.patron && destinatairesSurcharge.patronEmail) { adressesUniques.add(destinatairesSurcharge.patronEmail); }

if (destinatairesSurcharge.test && destinatairesSurcharge.test.trim() !== '') {

destinatairesSurcharge.test.split(',').map(e => e.trim()).forEach(email => adressesUniques.add(email));

}

} else {

if (config.Repondant\_Email\_Actif === 'Oui' && emailRepondantPrincipal) { adressesUniques.add(emailRepondantPrincipal); }

if (config.Patron\_Email\_Mode === 'Oui' && config.Patron\_Email) { adressesUniques.add(config.Patron\_Email); }

if (config.Formateur\_Email\_Actif === 'Oui' && config.Formateur\_Email) { adressesUniques.add(config.Formateur\_Email); }

}

if (config.Developpeur\_Email) { adressesUniques.add(config.Developpeur\_Email); }

adressesUniques.forEach(adresse => {

try {

let sujetFinal = sujet;

let corpsHtmlFinal = corpsHtml;

if (adresse.toLowerCase() !== (emailRepondantPrincipal || "").toLowerCase()) {

sujetFinal = (T.PREFIXE\_COPIE\_EMAIL || "Copie : ") + sujet;

if (contenuInfoCopie) { corpsHtmlFinal = contenuInfoCopie + corpsHtml; }

}

const mailOptions = { to: adresse, subject: sujetFinal, htmlBody: corpsHtmlFinal, attachments: piecesJointes };

const aliasExpediteur = optionsSurcharge.alias || config.Email\_Alias;

if (aliasExpediteur && aliasExpediteur.trim() !== '') {

mailOptions.from = aliasExpediteur;

}

GmailApp.sendEmail(mailOptions.to, mailOptions.subject, "", mailOptions);

Logger.log(`E-mail de RÉSULTATS [${langueCible}] envoyé avec succès à ${adresse}.`);

} catch (e) {

Logger.log(`Echec de l'envoi des résultats à ${adresse}. Erreur: ${e.message}`);

}

});

}

function getDonneesPourRetraitement(rowIndex) {

try {

const config = getTestConfiguration();

const reponse = \_creerObjetReponse(rowIndex);

const nomRepondant = reponse.Votre\_nom\_et\_prenom || reponse.Nom\_et\_prenom || '';

const emailRepondant = reponse.Votre\_adresse\_e\_mail || reponse.Adresse\_e\_mail || '';

return {

nomRepondant: nomRepondant,

emailRepondant: emailRepondant,

langueOrigine: getOriginalLanguage(reponse),

repondantActif: config.Repondant\_Email\_Actif === 'Oui',

formateurActif: config.Formateur\_Email\_Actif === 'Oui',

patronActif: config.Patron\_Email\_Mode === 'Oui',

emailAlias: config.Email\_Alias || ''

};

} catch (e) {

Logger.log(`ERREUR dans getDonneesPourRetraitement(${rowIndex}): ${e.toString()}`);

throw new Error("Impossible de récupérer les données pour la ligne " + rowIndex + ". " + e.message);

}

}

function lancerRetraitementDepuisUI(options) {

if (!options || !options.rowIndex) {

throw new Error("Les options de retraitement sont invalides.");

}

options.isRetraitement = true;

traiterLigne(options.rowIndex, options);

return "Retraitement pour la ligne " + options.rowIndex + " lancé avec succès !";

}

```

\*\*Fichier 2 : `Utilities.gs` (Version 8.2)\*\*

```javascript

// =================================================================================

// == FICHIER : Utilities.gs

// == VERSION : 8.2 (Robustesse de la détection de colonne de langue)

// == RÔLE : Boîte à outils du Kit de Traitement.

// =================================================================================

const ID\_FEUILE\_PILOTE = "1kLBqIHZWbHrb4SsoSQcyVsLOmqKHkhSA4FttM5hZtDQ";

/\*\*

\* Récupère la ligne de configuration complète pour le test en cours.

\*/

function getTestConfiguration() {

const ss = SpreadsheetApp.getActiveSpreadsheet();

const idSheetActuelle = ss.getId();

const piloteSheet = SpreadsheetApp.openById(ID\_FEUILE\_PILOTE);

const paramsSheet = piloteSheet.getSheetByName("Paramètres Généraux");

if (!paramsSheet) { throw new Error("L'onglet 'Paramètres Généraux' est introuvable."); }

const data = paramsSheet.getDataRange().getValues();

const headers = data.shift();

const idSheetColIndex = headers.indexOf('ID\_Sheet\_Cible');

if (idSheetColIndex === -1) { throw new Error("La colonne 'ID\_Sheet\_Cible' est introuvable."); }

const configRow = data.find(row => row[idSheetColIndex] === idSheetActuelle);

if (!configRow) { throw new Error("Impossible de trouver la configuration pour ce test (ID: " + idSheetActuelle + ").");}

const configuration = {};

headers.forEach((header, index) => {

if (header) { configuration[header] = configRow[index]; }

});

return configuration;

}

/\*\*

\* Lit l'onglet 'sys\_ID\_Fichiers' de la feuille de configuration centrale.

\*/

function getSystemIds() {

try {

const configSS = SpreadsheetApp.openById(ID\_FEUILE\_PILOTE);

const idSheet = configSS.getSheetByName('sys\_ID\_Fichiers');

if (!idSheet) { throw new Error("L'onglet 'sys\_ID\_Fichiers' est introuvable."); }

const data = idSheet.getDataRange().getValues();

const ids = {};

data.slice(1).forEach(row => {

if (row[0] && row[1]) { ids[row[0]] = row[1]; }

});

return ids;

} catch (e) {

Logger.log("Impossible de charger les ID système : " + e.toString());

throw new Error("Impossible de charger les ID système. Erreur: " + e.message);

}

}

/\*\*

\* Détecte correctement la langue de la réponse initiale de l'utilisateur.

\*/

function getOriginalLanguage(reponses) {

const langueRepondantBrute = reponses['Langue\_\_\_Language'] || reponses['Langue / Language'] || 'Français';

const mapLangue = { 'Français': 'FR', 'English': 'EN', 'Español': 'ES', 'Deutsch': 'DE' };

return mapLangue[langueRepondantBrute] || 'FR';

}

/\*\*

\* Charge les chaînes de caractères traduites pour une langue donnée.

\*/

function loadTraductions(langueCode) {

if (!langueCode) {

throw new Error("Le code de langue fourni à loadTraductions est indéfini.");

}

const systemIds = getSystemIds();

const bdd = SpreadsheetApp.openById(systemIds.ID\_BDD);

const traductionsSheet = bdd.getSheetByName("traductions");

if (!traductionsSheet) throw new Error("L'onglet 'traductions' est introuvable.");

const data = traductionsSheet.getDataRange().getValues();

const headers = data.shift();

const langColIndex = headers.findIndex(h => h && h.trim().toLowerCase() === langueCode.toLowerCase());

if (langColIndex === -1) throw new Error(`La colonne de langue '${langueCode}' est introuvable dans l'onglet "traductions".`);

const traductions = {};

const keyColIndex = 0;

data.forEach(row => {

if (row[keyColIndex]) { traductions[row[keyColIndex]] = row[langColIndex]; }

});

return traductions;

}

// NOTE: La fonction buildAndSendEmails et findAttachments sont considérées comme "legacy"

// et ne sont plus appelées par le moteur universel. Elles sont conservées pour référence.

function buildAndSendEmails(config, reponse, resultats, langueCode, isDebugMode, destinatairesSurcharge = {}) { /\* ... \*/ }

function findAttachments(typeTest, profilCode, niveauPJ, langueCode) { /\* ... \*/ }

```

\*\*Fichier 3 : `GestionTriggers.gs` (Version 1.0)\*\*

```javascript

// =================================================================================

// == FICHIER : GestionTriggers.gs

// == VERSION : 1.0 (Création initiale)

// == RÔLE : Gère la création et l'exécution des envois d'e-mails différés.

// =================================================================================

/\*\*

\* Calcule le délai en millisecondes à partir d'une chaîne de caractères (ex: "4h", "1j").

\*/

function \_calculerDelaiEnMs(valeurDelai) {

if (!valeurDelai || typeof valeurDelai !== 'string') return 0;

const valeurNumerique = parseInt(valeurDelai.replace(/[^0-9]/g, ''), 10);

if (isNaN(valeurNumerique)) return 0;

if (valeurDelai.includes('h')) {

return valeurNumerique \* 60 \* 60 \* 1000; // Heures en millisecondes

} else if (valeurDelai.includes('j')) {

return valeurNumerique \* 24 \* 60 \* 60 \* 1000; // Jours en millisecondes

} else if (valeurDelai.includes('min')) {

return valeurNumerique \* 60 \* 1000; // Minutes en millisecondes

}

return 0; // Format non reconnu

}

/\*\*

\* Programme l'envoi différé de l'e-mail de résultats.

\*/

function programmerEnvoiResultats(rowIndex, langueCible, delai) {

try {

const delaiEnMs = \_calculerDelaiEnMs(delai);

if (delaiEnMs <= 0) {

Logger.log(`Délai invalide ou nul (${delai}). Annulation de la programmation.`);

return;

}

const proprieteId = `envoiDiffere\_${new Date().getTime()}\_${rowIndex}`;

const properties = PropertiesService.getScriptProperties();

properties.setProperty(proprieteId, JSON.stringify({

rowIndex: rowIndex,

langueCible: langueCible

}));

ScriptApp.newTrigger('envoyerEmailProgramme')

.timeBased()

.after(delaiEnMs)

.create();

Logger.log(`Envoi programmé avec succès pour la ligne ${rowIndex}. Délai : ${delai}. ID de propriété : ${proprieteId}`);

} catch (e) {

Logger.log(`ERREUR lors de la programmation de l'envoi pour la ligne ${rowIndex}: ${e.toString()}\n${e.stack}`);

}

}

/\*\*

\* Fonction exécutée par le déclencheur pour envoyer l'e-mail de résultats.

\*/

function envoyerEmailProgramme(e) {

const properties = PropertiesService.getScriptProperties();

const toutesLesProps = properties.getProperties();

const proprieteId = Object.keys(toutesLesProps).find(key => key.startsWith('envoiDiffere\_'));

if (!proprieteId) {

Logger.log("Déclencheur d'envoi programmé exécuté, mais aucune propriété de tâche trouvée. Annulation.");

return;

}

try {

const donnees = JSON.parse(properties.getProperty(proprieteId));

const { rowIndex, langueCible } = donnees;

Logger.log(`Exécution de l'envoi programmé pour la ligne ${rowIndex} (ID: ${proprieteId})`);

const config = getTestConfiguration();

const reponse = \_creerObjetReponse(rowIndex);

const langueOrigine = getOriginalLanguage(reponse);

const resultats = calculerResultats(reponse, langueCible, config, langueOrigine);

assemblerEtEnvoyerEmailUniversel(config, reponse, resultats, langueCible, {});

properties.deleteProperty(proprieteId);

Logger.log(`Nettoyage de la propriété ${proprieteId} terminé.`);

} catch (err) {

Logger.log(`ERREUR FATALE lors de l'exécution de l'envoi programmé (ID: ${proprieteId}): ${err.toString()}\n${err.stack}`);

properties.deleteProperty(proprieteId);

} finally {

if (e && e.triggerUid) {

const allTriggers = ScriptApp.getProjectTriggers();

for (const trigger of allTriggers) {

if (trigger.getUniqueId() === e.triggerUid) {

ScriptApp.deleteTrigger(trigger);

Logger.log(`Déclencheur ${e.triggerUid} auto-détruit avec succès.`);

break;

}

}

}

}

}

```

\*(Note: Les autres fichiers comme `Logique\_Universel.gs`, `Menu.gs` etc. sont aussi inclus pour la complétude)\*

#### 5.2. Architecture Détaillée et Traçabilité

\* \*\*AJOUTÉ\*\* : Fichier `GestionTriggers.gs` (v1.0) - Isole toute la logique de programmation des envois différés.

\* \*\*MODIFIÉ\*\* : `TraitementReponses.gs` (v19.4) - Intégration de la logique de décision (immédiat vs. différé), ajout de la priorité au retraitement manuel, et ajout de robustesse sur la lecture des ID de pièces jointes (`.trim()`) et leur récupération (boucle de re-tentative).

\* \*\*MODIFIÉ\*\* : `Utilities.gs` (v8.2) - Amélioration de la fonction `loadTraductions` pour ignorer les espaces dans les en-têtes de colonnes de langue (`.trim()`).

#### 5.3. Nomenclature des Variables

\* \*\*Nom Précis\*\* : `Repondant\_Quand`

\* \*\*Type\*\* : `String`

\* \*\*Origine/Portée\*\* : Colonne dans `Paramètres Généraux`. Provient des options listées dans `sys\_Options\_Parametres`.

\* \*\*Contenu Attendu\*\* : Une chaîne de caractères comme "Immediat", "4h", "24h", "1j".

\* \*\*Exemple Concret\*\* : `"24h"`

\* \*\*Nom Précis\*\* : `envoiDiffere\_[timestamp]\_[rowIndex]`

\* \*\*Type\*\* : `String` (clé) / `String` (valeur JSON)

\* \*\*Origine/Portée\*\* : Propriété de script (`PropertiesService`), créée temporairement par `programmerEnvoiResultats`.

\* \*\*Contenu Attendu\*\* : La clé est un identifiant unique ; la valeur est un JSON contenant `rowIndex` et `langueCible`.

\* \*\*Exemple Concret\*\* : Clé=`"envoiDiffere\_1755829231084\_6"`, Valeur=`"{"rowIndex":6,"langueCible":"FR"}"`

#### 5.4. Structure des Données (Feuilles et Colonnes)

\* \*\*Fichier\*\* : `[CONFIG]V2 Usine à Tests`

\* \*\*Onglet Critique\*\* : `sys\_Options\_Parametres`

\* \*\*Colonnes Critiques\*\* : `Repondant\_Quand`, `Patron\_Quand`, etc. - Servent à peupler les listes déroulantes de l'interface de configuration.

\* \*\*Fichier\*\* : BDD (Base de Données)

\* \*\*Onglet Critique\*\* : `sys\_Composition\_Emails`

\* \*\*Colonnes Critiques\*\* : `Type\_Test`, `Code\_Langue`, `Code\_Profil`, `Element` (utilisés pour filtrer), `Contenu / ID\_Document` (contient l'ID de la pièce jointe à nettoyer avec `.trim()`).

---

### 6. Actions Recommandées / Prochaines Étapes

\* \*\*Action Immédiate (Pour Vous)\*\* :

1. Remplacer vos scripts par les versions finales listées ci-dessus.

2. Nettoyer les données de test (supprimer l'option `1min` de `sys\_Options\_Parametres`).

3. Configurer les délais souhaités pour vos tests dans `Paramètres Généraux`.

\* \*\*Questions pour notre prochaine session\*\* :

1. Pouvons-nous considérer la fonctionnalité "Envoi Différé" comme 100% validée et terminée ?

2. Quelle est la prochaine priorité de développement ? (Nouvelle fonctionnalité, nouveau test, etc.)

3. Souhaitez-vous que nous procédions au "nettoyage" du code inutilisé (`sys\_PiecesJointes` et la fonction `findAttachments`) pour plus de clarté ?

---

### Règle Impérative pour les Sessions Futures

Je note et conserverai cette instruction pour toutes nos interactions futures :

> règle pour la suite : Gemini fournira systématiquement la version complète et mise à jour de chaque script modifié pour éviter toute confusion.

J'ai bien lu cette règle et je m'engage à la respecter.