Absolument. Voici la synthèse complète de notre session, structurée selon votre modèle. Le document est prêt à être sauvegardé pour notre prochaine discussion.

Markdown

---

Projet: "Usine à Tests"

Date: "2508181732"

Version: "11.0"

---

### 1. Résumé des Échanges du Jour

La session d'aujourd'hui a été consacrée à l'implémentation et au débogage en profondeur de la fonctionnalité permettant de \*\*changer le niveau de résultat (N1, N2, N3)\*\* lors d'un retraitement.

Le développement a suivi plusieurs étapes de résolution de problèmes :

1. \*\*Mise à jour de l'interface :\*\* Le fichier `RetraitementUI.html` a été modifié pour inclure une liste déroulante permettant le choix du niveau.

2. \*\*Premier obstacle (Moteur Legacy) :\*\* Les tests ont révélé que le script exécutait l'ancien moteur "Legacy". Le diagnostic a montré qu'il manquait une colonne `Moteur\_Calcul` dans le fichier de configuration `[CONFIG]V2 Usine à Tests`. La solution a été d'ajouter cette colonne et de la renseigner avec la valeur "Universel".

3. \*\*Second obstacle (Données invisibles) :\*\* Malgré l'utilisation du bon moteur, les pièces jointes restaient absentes. Grâce à des versions de débogage avancées du script ("Super Espion" puis "Diagnostic Ultime"), nous avons prouvé que la logique du script était correcte mais que des \*\*incohérences de données invisibles\*\* (espaces superflus) dans les colonnes `Type\_Test`, `Code\_Profil` et `Element` de la base de données faisaient échouer les comparaisons.

4. \*\*Solution finale et robuste :\*\* Le script `TraitementReponses.gs` a été modifié pour utiliser la fonction `.trim()` sur toutes les comparaisons de chaînes de caractères critiques. Cette correction a rendu le moteur de filtrage insensible aux erreurs de saisie (espaces) et a permis de résoudre définitivement le problème.

5. \*\*Succès et clarification :\*\* Les tests finaux ont confirmé le bon fonctionnement de la fonctionnalité, avec et sans forçage de niveau, et la bonne gestion des pièces jointes.

6. \*\*Nouvelles demandes :\*\* En fin de session, nous avons clarifié que le champ "Ajouter une adresse de test" sert à ajouter des \*\*destinataires\*\* et non à changer l'\*\*envoyeur\*\*. Vous avez demandé à ce que soit ajoutée comme prochaine étape la possibilité d'\*\*envoyer des e-mails depuis un alias Gmail\*\* que vous avez déjà configuré.

### 2. État Actuel du Projet

\* \*\*Tâches réalisées :\*\*

\* Développement complet et stabilisation de la fonctionnalité de forçage de niveau (N1/N2/N3) dans l'interface de retraitement.

\* Mise à jour du moteur de filtrage pour le rendre robuste aux espaces superflus dans les données critiques (`Type\_Test`, `Code\_Profil`, `Element`).

\* Correction de la configuration du projet pour permettre la sélection explicite du moteur de calcul ("Universel" ou "Legacy").

\* \*\*Tâches en cours :\*\*

\* Aucune. La fonctionnalité est considérée comme terminée et stable.

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

1. \*\*Développement de l'envoi via alias :\*\* Programmer la possibilité de choisir un alias Gmail comme adresse d'expédition des e-mails.

2. \*\*Développements liés au test "R&K" :\*\* Entamer les travaux pour intégrer ce nouveau type de test.

3. \*\*Vérification des tests existants :\*\* S'assurer que les tests "MBTI" et "Ancres" fonctionnent correctement avec le moteur Universel et sa nouvelle logique robuste.

### 3. Contexte Historique du Projet

Ce projet vise à créer une plateforme centralisée et pilotée par les données ("Usine à Tests") pour automatiser le traitement et l'envoi des résultats de tests. Après avoir validé la logique de calcul des scores et d'assemblage des textes, nous nous sommes heurtés à des problèmes de pièces jointes. Le débogage de ces problèmes a mis en lumière une faiblesse majeure : la sensibilité du script à la qualité des données. La session d'aujourd'hui, bien que centrée sur l'ajout d'une nouvelle fonctionnalité, a eu pour effet principal de \*\*solidifier et de fiabiliser le cœur du moteur de traitement\*\*, le rendant beaucoup moins "fragile".

### 4. Orientations Stratégiques

La session renforce massivement le principe d'architecture fondamental du projet : \*\*"Zéro Convention Implicite"\*\*. Nous avons prouvé que l'on ne peut pas avoir comme convention implicite que "les données saisies dans le tableur seront toujours propres". Le code doit activement se protéger contre les incohérences de données. L'intégration de `.trim()` sur les comparaisons de `Type\_Test`, `Code\_Profil` et `Element` est une incarnation directe de ce principe, passant d'un script "fragile" à un script "robuste".

L'ajout de la fonctionnalité de forçage de niveau et la future fonctionnalité d'envoi via alias s'inscrivent dans l'objectif stratégique de rendre la plateforme plus \*\*flexible, professionnelle et puissante\*\* pour l'utilisateur final.

### 5. Architecture Technique & Ressources Clés

#### 5.1. État des Scripts et Fichiers

Voici la version finale et complète du script `TraitementReponses.gs` qui est maintenant stable et robuste.

```javascript

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

// == FICHIER : TraitementReponses.gs

// == VERSION : 11.0 (Correctif Final & Robuste)

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

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

/\*\*

\* 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}`);

}

}

/\*\*

\* COEUR LOGIQUE : Aiguille le traitement vers le moteur "Legacy" ou "Universel".

\*/

function traiterLigne(rowIndex, optionsSurcharge = {}) {

const isDebugMode = (typeof LoggerLib !== 'undefined' && LoggerLib.MODE\_DEBUG\_ACTIF === true);

if (isDebugMode) { LoggerLib.logStart(`Traitement Ligne ${rowIndex}`, optionsSurcharge); }

try {

const config = getTestConfiguration();

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) => { reponse[header] = rowValues[i] });

reponse.nomRepondant = reponse['Votre nom et prénom'] || 'Participant';

reponse.emailRepondant = reponse['Votre adresse e-mail'] || null;

const langueOrigine = getOriginalLanguage(reponse);

const langueCible = optionsSurcharge.langue || langueOrigine;

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

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

Logger.log("Moteur UNIVERSEL détecté pour la ligne " + rowIndex);

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

} else {

Logger.log("Moteur LEGACY détecté pour la ligne " + rowIndex);

buildAndSendEmails(config, reponse, resultats, langueCible, isDebugMode, optionsSurcharge.destinataires);

}

} catch (err) {

if (isDebugMode) { LoggerLib.logError("Erreur fatale dans traiterLigne", err); }

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

} finally {

if (isDebugMode) { LoggerLib.logEnd(); }

}

}

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

// == NOUVEAU MOTEUR UNIVERSEL

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

/\*\*

\* Lit la BDD, assemble et envoie l'e-mail selon les règles de sys\_Composition\_Emails.

\*/

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;

Logger.log(`INFO: Le niveau de résultat a été forcé à [${codeNiveauEmail}] depuis l'interface de retraitement.`);

}

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')

};

Logger.log(`--- DÉBUT DU FILTRAGE AVEC LES PARAMÈTRES SUIVANTS ---`);

Logger.log(`Type Test: [${typeTest}], Langue: [${langueCible}], Niveau: [${codeNiveauEmail}], Profil: [${profilFinal}]`);

Logger.log(`----------------------------------------------------`);

let briquesDeContenu = compoData.filter(row => {

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

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 === '');

if ((row[idx.element] || '').trim() === 'Document') {

Logger.log(`--- Vérification Ligne 'Document' (Ordre: ${row[idx.ordre]}) ---`);

Logger.log(`1. Type Test: [${row[idx.typeTest]}] vs [${typeTest}] -> ${typeMatch}`);

Logger.log(`2. Langue: [${row[idx.langue]}] vs [${langueCible}] -> ${langMatch}`);

Logger.log(`3. Niveau Email: [${row[idx.niveau]}] inclut [${codeNiveauEmail}] ? -> ${levelMatch}`);

Logger.log(`4. Profil: [${row[idx.profil]}] vs [${profilFinal}] ? -> ${profileMatch}`);

Logger.log('----------------------------------------------------');

}

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

});

Logger.log(`FILTRAGE TERMINÉ. Nombre total de briques sélectionnées : ${briquesDeContenu.length}`);

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

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) piecesJointesIds.add(contenu); 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;

default: break;

}

}

sujet = sujet.replace(/{{nom\_repondant}}/g, reponse.nomRepondant).replace(/{{profil\_titre}}/g, resultats.titreProfil);

corpsHtml = corpsHtml.replace(/{{nom\_repondant}}/g, reponse.nomRepondant).replace(/{{profil\_titre}}/g, resultats.titreProfil).replace(/{{profil\_description}}/g, resultats.descriptionProfil);

const piecesJointes = [];

piecesJointesIds.forEach(id => {

try {

piecesJointes.push(DriveApp.getFileById(id).getBlob());

} catch (e) {

Logger.log(`ERREUR lors de la récupération du fichier Drive avec l'ID : ${id}. Détails : ${e.message}`);

}

});

const T = loadTraductions(langueCible);

const adressesUniques = new Set();

const destinatairesSurcharge = optionsSurcharge.destinataires || {};

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

if (useSurcharge) {

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

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' && reponse.emailRepondant) { adressesUniques.add(reponse.emailRepondant); }

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;

if (adresse.toLowerCase() !== (reponse.emailRepondant || "").toLowerCase()) {

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

}

MailApp.sendEmail({ to: adresse, subject: sujetFinal, htmlBody: corpsHtml, attachments: piecesJointes });

} catch (e) {

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

}

});

}

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

// == SECTION INTERFACE UTILISATEUR (UI)

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

function retraiterReponse\_UI() {

const ui = SpreadsheetApp.getUi();

const htmlOutput = HtmlService.createHtmlOutputFromFile('DialogueLigne.html')

.setWidth(350)

.setHeight(160);

ui.showModalDialog(htmlOutput, 'Retraitement de Réponse');

}

function ouvrirSidebarPourLigne(rowIndex) {

const ui = SpreadsheetApp.getUi();

const template = HtmlService.createTemplateFromFile('RetraitementUI');

template.ligneActive = rowIndex;

const htmlOutput = template.evaluate()

.setTitle("Retraitement - Ligne " + rowIndex)

.setWidth(350);

ui.showSidebar(htmlOutput);

}

function getDonneesPourRetraitement(rowIndex) {

try {

const config = getTestConfiguration();

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

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

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

const reponse = {};

headers.forEach((h, i) => { reponse[h] = values[i]; });

return {

nomRepondant: reponse['Votre nom et prénom'] || '',

emailRepondant: reponse['Votre adresse e-mail'] || '',

langueOrigine: getOriginalLanguage(reponse),

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

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

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

};

} 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.");

}

traiterLigne(options.rowIndex, {

langue: options.langue,

niveau: options.niveau,

destinataires: options.destinataires

});

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

}

**5.2. Architecture Détaillée et Traçabilité**

* **MODIFIÉ :** TraitementReponses.gs (Version 11.0 "Correctif Final & Robuste"). Le moteur de filtrage a été rendu insensible aux espaces dans les colonnes Type\_Test, Code\_Profil et Element. La logique d'assemblage des pièces jointes a également été robustifiée. Le "Super Espion" a été amélioré.
* **MODIFIÉ :** [CONFIG]V2 Usine à Tests. Ajout de la colonne Moteur\_Calcul pour permettre la sélection explicite du moteur de traitement pour chaque test.
* **MODIFIÉ :** RetraitementUI.html. Ajout d'une liste déroulante pour forcer le niveau de résultat lors d'un retraitement.
* **VÉRIFIÉ :** [BDD]V2 Tests & Profils. Les données de l'onglet sys\_Composition\_Emails ont été corrigées et vérifiées.

**5.3. Nomenclature des Variables**

* typeMatch (Boolean) : Résultat de la comparaison entre le Type\_Test du test et celui de la BDD.
* levelMatch (Boolean) : Résultat de la vérification includes() entre le niveau du test et celui de la BDD.
* profileMatch (Boolean) : Résultat de la comparaison entre le profilFinal du participant et le Code\_Profil de la BDD (ou si ce dernier est vide).
* elementType (String) : Contient la valeur "nettoyée" (.trim()) de la colonne Element avant le switch pour fiabiliser la décision.

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

* Fichier [CONFIG]V2 Usine à Tests : La colonne Moteur\_Calcul a été ajoutée. Les valeurs attendues sont "Universel" ou "Legacy" (ou vide pour Legacy).
* Fichier [BDD]V2 Tests & Profils (onglet sys\_Composition\_Emails) : Une attention critique a été portée aux colonnes Type\_Test, Code\_Niveau\_Email, Code\_Profil et Element qui pilotent l'intégralité du filtrage.

**6. Actions Recommandées / Prochaines Étapes**

* **Action Immédiate (Pour Vous) :**
  1. Pour les tests "MBTI" et "Ancres", vérifiez que la colonne Moteur\_Calcul est bien sur **"Universel"** dans le fichier de configuration si vous souhaitez qu'ils bénéficient du nouveau moteur robuste.
  2. Continuez à configurer vos tests et leurs documents associés dans la BDD en suivant la logique de niveaux que nous avons validée.
* **Questions pour notre prochaine session :**
  1. **Envoi par Alias :** Comment souhaitez-vous spécifier l'alias à utiliser ? Est-ce une option globale pour tous les tests, ou un paramètre à définir par test dans le fichier de configuration ?
  2. **Test "R&K" :** Pour préparer son développement, pourriez-vous décrire sa logique ? A-t-il des profils, des niveaux de résultats, des pièces jointes spécifiques ?