Absolument. Ce fut une session de travail extrêmement dense et productive. Nous avons non seulement résolu des bugs complexes, mais nous avons aussi amélioré nos propres méthodes de travail pour l'avenir.

Conformément à vos instructions, voici la synthèse complète de notre session, prête à être sauvegardée.

Markdown

---

Projet: "Usine à Tests"

Date: "2025-07-21"

Version: "7.0"

---

### 1. Résumé des Échanges du Jour

La session d'aujourd'hui a été un marathon de développement et de débogage en profondeur, aboutissant à la stabilisation complète de la chaîne de traitement et à une amélioration de notre workflow de collaboration.

Le processus a suivi plusieurs étapes clés :

\* \*\*Finalisation de l'Interface de Retraitement :\*\* Nous avons repris le projet pour construire l'interface de retraitement des réponses. Nous avons opté pour une barre latérale (`sidebar`) pour sa flexibilité future et avons développé les scripts nécessaires (`TraitementReponses.gs` v4.1, `RetraitementUI.html`).

\* \*\*Débogage Approfondi du Moteur Multilingue :\*\* Un test a révélé un bug critique où les scores n'étaient pas calculés pour les e-mails en anglais. Ce débogage nous a amenés à inspecter et valider toute la chaîne de données :

1. `traductions` (OK)

2. `Gabarits\_Emails` (OK)

3. `Profils\_Couleurs\_EN` (OK)

4. Incohérence `Type` / `mode` JSON (Corrigé)

5. \*\*Cause Racine :\*\* Nous avons finalement découvert que les `libelle` (textes des réponses) dans le JSON de la feuille `Questions\_Couleurs\_EN` n'avaient pas été traduits, empêchant le calcul des scores.

\* \*\*Correction du Script de Configuration :\*\* Après avoir résolu le bug de calcul, un nouveau test a révélé que la colonne `ID\_Gabarit\_Email\_Repondant` n'était pas remplie lors de la création d'un test. Nous avons diagnostiqué que le script `menu.gs` de `[CONFIG] Usine à Tests` ne traduisait pas le choix de l'utilisateur ("Niveau de contenu") en un ID de gabarit. Nous avons corrigé ce script pour combler cette lacune.

\* \*\*Mise en Place d'un Nouveau Workflow :\*\* Face à la complexité du projet et aux limites de ma "mémoire" entre les sessions, nous avons validé une nouvelle méthode de travail. Chaque session commencera par un "Kit de Démarrage" fourni par vos soins, et nous allons construire un \*\*"Dictionnaire des Données"\*\* pour figer la nomenclature du projet.

### 2. État Actuel du Projet

\* \*\*Tâches réalisées :\*\*

\* Développement et validation de l'interface de retraitement des réponses (sidebar).

\* Identification et correction du bug critique lié au calcul des scores en environnement multilingue.

\* Correction du script `menu.gs` dans `[CONFIG] Usine à Tests` pour assurer le remplissage correct de la colonne `ID\_Gabarit\_Email\_Repondant`.

\* Définition d'un workflow de collaboration robuste (Kit de Démarrage, Dictionnaire des Données).

\* \*\*Tâches en cours :\*\*

\* Génération de ce document de synthèse.

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

\* Construire la première version du "Dictionnaire des Données" à partir des informations de structure que vous allez me fournir.

\* Enrichir l'interface de retraitement avec de nouvelles fonctionnalités (ex: choix direct du gabarit, surcharge des destinataires).

\* Mettre à jour le `Manuel d'Architecture` et le `Manuel Utilisateur` pour refléter les derniers changements.

### 3. Contexte Historique du Projet

La session d'aujourd'hui s'inscrit dans la phase de maturation de l'architecture V2 de l'Usine Universelle. Alors que nous pensions les principaux composants stables, une série de tests en conditions réelles a révélé des bugs subtils liés à la gestion des états et des données multilingues. Cette session a donc été une validation en profondeur qui nous a non seulement forcés à corriger ces bugs, mais surtout à améliorer nos propres processus de développement en instaurant un système de contexte partagé (`Kit de Démarrage`) et une source de vérité pour la nomenclature (`Dictionnaire des Données`), marquant une étape significative vers la maintenabilité à long terme du projet.

### 4. Orientations Stratégiques

Nos principes directeurs ont été au cœur de cette session de débogage intense :

\* \*\*Zéro Convention Implicite :\*\* Ce principe a été validé à de multiples reprises. Chaque bug résolu provenait d'une convention implicite : que les `libelle` dans le JSON seraient les mêmes, que le `Type` et le `mode` seraient cohérents, ou qu'une colonne de configuration serait remplie. Nos corrections ont consisté à rendre ces liens explicites. La décision de créer un `Dictionnaire des Données` est l'application ultime de ce principe à notre propre collaboration.

\* \*\*Le Code plus Intelligent que la Configuration :\*\* La correction apportée au `menu.gs` pour qu'il "comprenne" que "Niveau 2 : Détaillé" signifie `RESULTATS\_N2` est une amélioration directe de ce principe.

### 5. Architecture Technique & Ressources Clés

#### 5.1. État des Scripts et Fichiers

Voici le script `menu.gs` du projet `[CONFIG]V2 Usine à Tests` que nous avons corrigé aujourd'hui. C'est la version finale et complète.

```javascript

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

// == FICHIER : menu.gs (Version Corrigée)

// == VERSION : 3.2 (Correction du transfert de l'ID Gabarit)

// == RÔLE : Logique côté serveur pour l'application web de configuration.

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

const ID\_FEUILLE\_CONFIG = "1kLBqIHZWbHrb4SsoSQcyVsLOmqKHkhSA4FttM5hZtDQ";

// --- SECTION 1 : INTERFACE UTILISATEUR ---

function onOpen() {

SpreadsheetApp.getUi()

.createMenu('🚀 Actions Usine')

.addItem('Configurer un nouveau test...', 'showConfigurationSidebar')

.addToUi();

}

function showConfigurationSidebar() {

const html = HtmlService.createHtmlOutputFromFile('FormulaireUI')

.setTitle('Configuration Usine à Tests')

.setWidth(400);

SpreadsheetApp.getUi().showSidebar(html);

}

// --- SECTION 2 : FONCTIONS APPELÉES PAR L'INTERFACE HTML ---

function getInitialData() {

const ss = SpreadsheetApp.openById(ID\_FEUILLE\_CONFIG);

const optionsSheet = ss.getSheetByName("sys\_Options\_Parametres");

if (!optionsSheet) {

throw new Error("L'onglet 'sys\_Options\_Parametres' est introuvable.");

}

const optionsData = optionsSheet.getDataRange().getValues();

const headers = optionsData.shift();

const optionsMap = {};

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

const options = optionsData.map(row => row[i]).filter(String);

optionsMap[header] = options;

});

// Charger la liste des blocs méta disponibles depuis la BDD

let availableMetaBlocks = [];

try {

const systemIds = getSystemIds();

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

const metaSheet = bdd.getSheetByName('Questions\_META\_FR');

if (metaSheet) {

const metaData = metaSheet.getRange(2, 1, metaSheet.getLastRow() - 1, 3).getValues(); // ID, Type, Titre

availableMetaBlocks = metaData.map(row => ({ id: row[0], title: row[2] })).filter(block => block.id && block.title);

}

} catch(e) {

console.error("Impossible de charger les blocs méta depuis la BDD : " + e.message);

}

return {

typesDeTest: optionsMap['Type\_Test'] || [],

availableMetaBlocks: availableMetaBlocks,

options: {

Repondant\_Quand: optionsMap['Repondant\_Quand'] || [],

Repondant\_Contenu: optionsMap['Repondant\_Contenu'] || [],

Patron\_Quand: optionsMap['Patron\_Quand'] || [],

Patron\_Contenu: optionsMap['Patron\_Contenu'] || [],

Formateur\_Quand: optionsMap['Formateur\_Quand'] || [],

Formateur\_Contenu: optionsMap['Formateur\_Contenu'] || []

}

};

}

function getQuestionCountForTestType(typeTest) {

if (!typeTest) return 0;

try {

const systemIds = getSystemIds();

if (systemIds && systemIds.ID\_BDD) {

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

const questionSheet = bdd.getSheets().find(s => s.getName().startsWith('Questions\_' + typeTest));

if (questionSheet) {

return questionSheet.getLastRow() - 1;

}

}

return 0;

} catch (err) {

Logger.log('Erreur lors du calcul du nombre de questions pour ' + typeTest + ': ' + err.message);

return 0;

}

}

// --- SECTION 3 : TRAITEMENT DE LA SOUMISSION ---

function processNewTestConfiguration(formObject) {

try {

const ss = SpreadsheetApp.openById(ID\_FEUILLE\_CONFIG);

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

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

// S'assurer que les nouvelles colonnes existent avant de continuer

let headers = paramsSheet.getRange(1, 1, 1, paramsSheet.getLastColumn()).getValues()[0];

const requiredHeaders = ['Blocs\_Meta\_A\_Inclure', 'ID\_Gabarit\_Email\_Repondant'];

requiredHeaders.forEach(headerName => {

if (headers.indexOf(headerName) === -1) {

paramsSheet.getRange(1, paramsSheet.getLastColumn() + 1).setValue(headerName);

}

});

// Re-lire les en-têtes au cas où ils auraient été modifiés

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

let emailDev = formObject.devEmail;

if (!emailDev || emailDev.trim() === "") { emailDev = "chanenam@gmail.com"; }

const limiteLignes = getQuestionCountForTestType(formObject.type);

const blocsMetaString = formObject.blocsMeta.join(',');

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

// == DÉBUT DE LA CORRECTION : Traduire le "Niveau de contenu" en ID Gabarit

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

let idGabaritRepondant = ''; // Valeur par défaut : vide

if (formObject.repondantContenu === 'Niveau 1 : Simple') {

idGabaritRepondant = 'RESULTATS\_N1';

} else if (formObject.repondantContenu === 'Niveau 2 : Détaillé') {

idGabaritRepondant = 'RESULTATS\_N2';

}

// Ajoutez d'autres "else if" si vous créez plus de niveaux/gabarits.

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

// == FIN DE LA CORRECTION

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

const dataRow = {

'Id\_Unique': '',

'Titre\_Formulaire\_Utilisateur': formObject.titre,

'Nom\_Fichier\_Complet': '',

'Statut': 'En construction',

'Type\_Test': formObject.type,

'Blocs\_Meta\_A\_Inclure': blocsMetaString,

'ID\_Gabarit\_Email\_Repondant': idGabaritRepondant, // <-- AJOUT de la donnée traduite

'ID\_Dossier\_Cible': '',

'Limite\_Lignes\_A\_Traiter': limiteLignes,

'nbQuestions': formObject.nbQuestions,

'Repondant\_Email\_Actif': formObject.repondantActif ? "Oui" : "Non",

'Repondant\_Quand': formObject.repondantQuand,

'Repondant\_Contenu': formObject.repondantContenu,

'Patron\_Email\_Mode': formObject.patronActif ? "Oui" : "Non",

'Patron\_Quand': formObject.patronQuand,

'Patron\_Contenu': formObject.patronContenu,

'Patron\_Email': formObject.patronEmail,

'Formateur\_Email\_Actif': formObject.formateurActif ? "Oui" : "Non",

'Formateur\_Quand': formObject.formateurQuand,

'Formateur\_Contenu': formObject.formateurContenu,

'Formateur\_Email': formObject.formateurEmail,

'Developpeur\_Email': emailDev,

'ID\_Formulaire\_Cible': '',

'ID\_Sheet\_Cible': ''

};

const nouvelleLigne = headers.map(header => dataRow[header] !== undefined ? dataRow[header] : '');

paramsSheet.appendRow(nouvelleLigne);

return "Configuration enregistrée avec succès !";

} catch (e) {

Logger.log("ERREUR lors de la sauvegarde de la configuration: " + e.toString());

throw new Error("Une erreur interne est survenue lors de la sauvegarde. " + e.message);

}

}

// --- SECTION 4 : FONCTIONS UTILITAIRES ---

function getSystemIds() {

const configSS = SpreadsheetApp.openById(ID\_FEUILLE\_CONFIG);

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;

}

**5.2. Architecture Détaillée et Traçabilité**

* **[CONFIG] Usine à Tests** (Projet Apps Script)
  + menu.gs (**MODIFIÉ**) : La fonction processNewTestConfiguration a été corrigée pour traduire le repondantContenu (ex: 'Niveau 2 : Détaillé') en un ID\_Gabarit\_Email\_Repondant (ex: 'RESULTATS\_N2') et l'écrire dans la feuille Paramètres Généraux.
  + FormulaireUI.html (INCHANGÉ) : L'interface de configuration des nouveaux tests.
* **[TEMPLATE] Kit de Traitement V2** (Projet Apps Script)
  + TraitementReponses.gs (**MODIFIÉ**) : La fonction retraiterReponse\_UI a été modifiée pour ne plus dépendre de la sélection de la cellule, mais pour demander le numéro de la ligne via un prompt avant d'afficher la sidebar.
  + Utilities.gs (**MODIFIÉ**) : La fonction buildAndSendEmails a été rendue plus robuste pour utiliser un gabarit par défaut (RESULTATS\_N1) si la configuration ID\_Gabarit\_Email\_Repondant est vide.
  + RetraitementUI.html (**AJOUTÉ**) : Nouvelle interface (sidebar) pour le retraitement des réponses.

**5.3. Nomenclature des Variables**

*Ébauche de notre Dictionnaire des Données. À compléter lors de la prochaine session.*

| Type d'Élément | Nom Exact | Rôle |
| --- | --- | --- |
| Colonne | ID\_Gabarit\_Email\_Repondant | **(dans [CONFIG])** Spécifie l'ID du gabarit d'e-mail à utiliser pour le répondant. |
| Colonne | ID\_Gabarit | **(dans [BDD])** Clé primaire unique pour un modèle d'e-mail dans Gabarits\_Emails. |
| Colonne | Repondant\_Contenu | **(dans [CONFIG])** Choix de l'utilisateur dans l'UI de configuration (ex: "Niveau 2 : Détaillé"). |

Exporter vers Sheets

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

* **Fichier : [CONFIG]V2 Usine à Tests**
  + **Onglet Paramètres Généraux :** ID\_Gabarit\_Email\_Repondant, Id\_Unique, Titre\_Formulaire\_Utilisateur, Nom\_Fichier\_Complet, Statut, Lien\_Formulaire\_Public, Type\_Test, Blocs\_Meta\_A\_Inclure, ID\_Dossier\_Cible, Limite\_Lignes\_A\_Traiter, nbQuestions, Repondant\_Quand, Patron\_Email\_Mode, Patron\_Quand, Patron\_Contenu, Patron\_Email, Formateur\_Email\_Actif, Formateur\_Nom, Formateur\_Quand, Formateur\_Contenu, Formateur\_Email, Developpeur\_Email, ID\_Formulaire\_Cible, ID\_Sheet\_Cible.
  + **Onglet sys\_Options\_Parametres :** Statut, Type\_Test, Repondant\_Email\_Actif, Repondant\_Quand, Repondant\_Contenu, Patron\_Email\_Mode, Patron\_Quand, Patron\_Contenu, Formateur\_Email\_Actif, Formateur\_Quand, Formateur\_Contenu.

**6. Actions Recommandées / Prochaines Étapes**

* **Action Immédiate :** Me fournir la liste des en-têtes de colonnes pour les onglets restants de la [BDD] (Gabarits\_Emails, traductions, sys\_PiecesJointes, un exemple de Questions\_... et Profils\_...) afin de finaliser notre Dictionnaire des Données.
* **Prochaine Session :**
  1. Démarrer la session en utilisant notre nouveau workflow "Kit de Démarrage".
  2. Valider la première version complète du Dictionnaire des Données.
  3. Discuter et implémenter la prochaine fonctionnalité pour l'interface de retraitement : permettre à l'utilisateur de surcharger les destinataires de l'e-mail (répondant, formateur, etc.) et d'ajouter une adresse de test.

---

Projet: "Usine à Tests"

Date: "2025-07-21"

Version: "8.0"

---

### 1. Résumé des Échanges du Jour

La session d'aujourd'hui a marqué la \*\*finalisation et la stabilisation\*\* de plusieurs fonctionnalités critiques, ainsi que la mise en place d'une méthode de travail structurée pour l'avenir.

\* \*\*Finalisation du "Dictionnaire des Données"\*\* : Nous avons terminé la collecte des informations de structure (noms d'onglets et de colonnes) pour les fichiers maîtres `[CONFIG]V2 Usine à Tests` et `[BDD]V2 Tests & Profils`. J'ai ensuite généré la première version de ce dictionnaire, qui servira de référence officielle pour toute la nomenclature du projet.

\* \*\*Adoption d'un Workflow Structuré\*\* : Nous avons validé la méthode du \*\*"Kit de Démarrage"\*\*. Vous me fournirez au début de chaque session les scripts et documents pertinents pour assurer une continuité et une efficacité maximales. Le format de concaténation de scripts que vous avez développé a été approuvé comme étant idéal pour cet usage.

\* \*\*Relecture et Planification\*\* : Nous avons également mis à jour les documents clés du projet, tels que le \*\*Manuel d'Architecture\*\* et le \*\*Manuel Utilisateur\*\*, pour refléter les dernières évolutions (interface de retraitement, système d'e-mails dynamiques, etc.).

### 2. État Actuel du Projet

Le projet a atteint une phase de maturité fonctionnelle. Les principaux chantiers de développement sont terminés, et nous entrons dans une phase de consolidation et d'enrichissement.

\* \*\*Tâches réalisées :\*\*

\* Finalisation de la collecte d'informations pour la nomenclature.

\* Génération de la version 1.0 du "Dictionnaire des Données".

\* Validation du format de transmission des scripts via un fichier concaténé.

\* Mise à jour de la documentation projet (`Manuel d'Architecture` v3.0, `Manuel Utilisateur` v3.0).

\* \*\*Tâches en cours :\*\*

\* Génération de ce document de synthèse.

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

\* Intégrer le "Dictionnaire des Données" comme document de référence officiel du projet.

\* Commencer l'enrichissement de l'interface de retraitement (`RetraitementUI.html`) en y ajoutant la possibilité de surcharger les destinataires de l'e-mail.

### 3. Contexte Historique du Projet

Cette session conclut une longue phase de débogage et de stabilisation. Après avoir résolu des problèmes complexes dans la logique multilingue et dans la chaîne de configuration, nous avons capitalisé sur ces apprentissages pour renforcer nos propres processus de développement. La création du "Dictionnaire des Données" et l'adoption d'un "Kit de Démarrage" sont des réponses directes aux difficultés rencontrées, visant à prévenir la récurrence de bugs liés à des incohérences de nomenclature ou à une perte de contexte.

### 4. Orientations Stratégiques

La création du \*\*"Dictionnaire des Données"\*\* est l'application la plus aboutie de notre principe directeur \*\*"Zéro Convention Implicite"\*\*. En rendant chaque nom de colonne et d'onglet explicite et en le documentant, nous éliminons une source majeure d'erreurs potentielles. Cette session a donc transformé un principe de développement de code en un principe de gestion de projet.

### 5. Architecture Technique & Ressources Clés

#### 5.1. État des Scripts et Fichiers

Aucun script n'a été modifié durant cette session. Les dernières versions validées sont celles de notre précédente session de débogage. Le principal livrable de cette session est le Dictionnaire des Données ci-dessous.

#### 5.2. Architecture Détaillée et Traçabilité

Aucun changement dans l'architecture des scripts. L'architecture est considérée comme stable.

#### 5.3. Nomenclature des Variables et Structure des Données (Dictionnaire v1.0)

Voici la première version complète de notre Dictionnaire des Données.

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

\* \*\*Onglet `Paramètres Généraux`\*\*

\* `ID\_Gabarit\_Email\_Repondant`: Spécifie l'ID du modèle d'e-mail à utiliser.

\* `Id\_Unique`: Identifiant interne unique du test.

\* `Titre\_Formulaire\_Utilisateur`: Titre du questionnaire vu par les participants.

\* `Nom\_Fichier\_Complet`: Nom des fichiers (Form + Sheet) générés.

\* `Statut`: État d'avancement du test ("En construction", "Actif").

\* `Lien\_Formulaire\_Public`: URL publique du Google Form.

\* `Type\_Test`: Code du test (ex: "Couleurs") liant à la BDD.

\* `Blocs\_Meta\_A\_Inclure`: Liste des questions standards à inclure (ex: `META\_EMAIL`).

\* `ID\_Dossier\_Cible`: ID du dossier Drive de destination.

\* `nbQuestions`: Nombre de questions du test à utiliser.

\* `Repondant\_Email\_Actif`: "Oui"/"Non", active l'envoi d'e-mail au répondant.

\* `Repondant\_Quand`: Condition d'envoi de l'e-mail.

\* `Repondant\_Contenu`: Choix de l'utilisateur dans l'UI (ex: "Niveau 2 : Détaillé").

\* `Patron\_Email\_Mode`, `Patron\_Quand`, `Patron\_Contenu`, `Patron\_Email`: Options pour l'e-mail du patron.

\* `Formateur\_Email\_Actif`, `Formateur\_Nom`, `Formateur\_Quand`, `Formateur\_Contenu`, `Formateur\_Email`: Options pour l'e-mail du formateur.

\* `Developpeur\_Email`: Adresse e-mail pour les notifications techniques.

\* `ID\_Formulaire\_Cible`, `ID\_Sheet\_Cible`: ID des fichiers générés.

\* \*\*Onglet `sys\_Options\_Parametres`\*\*

\* Rôle : Alimente les menus déroulants de l'interface de configuration.

\* \*\*Onglet `sys\_ID\_Fichiers`\*\*

\* Rôle : Table de correspondance pour les ID des fichiers système (BDD, MOTEUR, etc.).

##### \*\*Fichier : `[BDD]V2 Tests & Profils`\*\*

\* \*\*Onglet `Questions\_[Type]\_[Langue]`\*\*

\* `ID`: Identifiant unique de la question.

\* `TypeQuestion`: Type de la question (ex: `QRM\_CAT`), doit être cohérent avec le `mode` JSON.

\* `Titre`, `Options`, `Logique`, `Description`: Champs descriptifs de la question.

\* `Paramètres (JSON)`: Contient la logique de calcul moderne.

\* \*\*Onglet `Profils\_[Type]\_[Langue]`\*\*

\* Rôle : Stocke les textes (Nom, Titre, Description) pour chaque profil de résultat.

\* \*\*Onglet `Traductions`\*\*

\* Rôle : Centralise les chaînes de texte statiques. (Colonnes: `Clé`, `FR`, `EN`...).

\* \*\*Onglet `sys\_PiecesJointes`\*\*

\* Rôle : Définit les fichiers à joindre aux e-mails.

\* \*\*Onglet `Gabarits\_Emails`\*\*

\* Rôle : Contient les modèles complets pour les e-mails dynamiques. (Colonnes: `ID\_Gabarit`, `Langue`, `Sujet`, `Corps\_...`, `Niveau\_Details\_Resultats`, etc.).

### 6. Actions Recommandées / Prochaines Étapes

\* \*\*Action Immédiate :\*\* Adopter officiellement le nouveau workflow. Pour notre prochaine session, me fournir un "Kit de Démarrage" incluant ce document de synthèse, le `Manuel d'Architecture` v3.0, et un fichier concaténé des scripts que nous prévoyons de modifier.

\* \*\*Prochaine Session :\*\*

1. Commencer le développement de la prochaine fonctionnalité : l'enrichissement de l'interface de retraitement.

2. Objectif : Ajouter des champs dans `RetraitementUI.html` et la logique dans `TraitementReponses.gs` pour permettre de surcharger les destinataires de l'e-mail (cocher/décocher le répondant, le formateur, et ajouter une adresse de test).

// Fichier généré le 21/07/2025 à 13:09:11

// --- Début du fichier: .clasp.json ---

{

"scriptId": "14jdOLhU0GVF3RLINBgZPLikLPSQBQv7w6aJYMmeoAG0w0UUL\_Bm6bIjx",

"rootDir": "",

"scriptExtensions": [

".js",

".gs"

],

"htmlExtensions": [

".html"

],

"jsonExtensions": [

".json"

],

"filePushOrder": [],

"skipSubdirectories": false

}

// --- Fin du fichier: .clasp.json ---

// --- Début du fichier: appsscript.json ---

{

"timeZone": "Indian/Mauritius",

"dependencies": {

},

"exceptionLogging": "STACKDRIVER",

"runtimeVersion": "V8"

}

// --- Fin du fichier: appsscript.json ---

// --- Début du fichier: concat\_scripts\_rep.js ---

const fs = require('fs');

const path = require('path');

// --- CONFIGURATION ---

// Extensions de fichiers à inclure

const scriptExtensions = [

'.js', '.ts', '.jsx', '.tsx', '.py', '.html', '.css', '.scss', '.less',

'.json', '.xml', '.php', '.rb', '.java', '.c', '.cpp', '.cs', '.go',

'.sh', '.ps1', '.bat', '.cmd', '.sql', '.vue', '.svelte', '.astro'

];

// Le dossier à scanner

const folderToScan = './'; // À ajuster si besoin

// --- FIN CONFIGURATION ---

// Récupérer le nom du répertoire scanné

const resolvedFolderPath = path.resolve(folderToScan);

const dirName = path.basename(resolvedFolderPath);

// Créer une date au format aammjj

const now = new Date();

const year = String(now.getFullYear()).slice(2); // 2 derniers chiffres de l'année

const month = String(now.getMonth() + 1).padStart(2, '0');

const day = String(now.getDate()).padStart(2, '0');

const dateStamp = `${year}${month}${day}`;

// Nom du fichier de sortie

const outputFileName = `${dirName}\_Script\_${dateStamp}.txt`;

// Créer une chaîne avec date et heure en français

const dateTimeString = `// Fichier généré le ${now.toLocaleDateString('fr-FR')} à ${now.toLocaleTimeString('fr-FR')}\n\n`;

// Initialiser la variable qui accumulera le contenu

let allContent = dateTimeString;

// Fonction récursive

function readFilesRecursively(directory) {

fs.readdirSync(directory).forEach(file => {

const absolutePath = path.join(directory, file);

if (fs.statSync(absolutePath).isDirectory()) {

readFilesRecursively(absolutePath);

} else {

const fileExtension = path.extname(file).toLowerCase();

if (scriptExtensions.includes(fileExtension)) {

allContent += `// --- Début du fichier: ${absolutePath} ---\n`;

allContent += fs.readFileSync(absolutePath, 'utf8');

allContent += `\n// --- Fin du fichier: ${absolutePath} ---\n\n`;

}

}

});

}

// Supprimer le fichier de sortie existant

if (fs.existsSync(outputFileName)) {

fs.unlinkSync(outputFileName);

console.log(`Ancien fichier '${outputFileName}' supprimé.`);

}

// Lancer le traitement

try {

readFilesRecursively(folderToScan);

fs.writeFileSync(outputFileName, allContent, 'utf8');

console.log(`Succès : Tous les scripts ont été exportés dans '${outputFileName}'`);

} catch (error) {

console.error(`Erreur lors de la lecture ou de l'écriture des fichiers : ${error.message}`);

}

// --- Fin du fichier: concat\_scripts\_rep.js ---

// --- Début du fichier: FormulaireUI.html ---

<!-- ================================================================================= -->

<!-- == FICHIER : FormulaireUI.html == -->

<!-- == Rôle : Interface utilisateur dynamique pour la configuration des tests. == -->

<!-- == VERSION : 3.1 - Correction de l'affichage du nombre de questions == -->

<!-- ================================================================================= -->

<!DOCTYPE html>

<html>

  <head>

    <base target="\_top">

    <script src="https://cdn.tailwindcss.com"></script>

    <style>

      .hidden-section { max-height: 0; overflow: hidden; transition: max-height 0.5s ease-out, opacity 0.3s ease-out; opacity: 0; }

      .visible-section { max-height: 500px; opacity: 1; transition: max-height 0.5s ease-in, opacity 0.5s ease-in; }

    </style>

  </head>

  <body class="bg-gray-100 p-4 font-sans">

    <div id="main-form">

      <h1 class="text-xl font-bold text-gray-800 mb-4">Configurer un Nouveau Test</h1>

      <!-- Titre du Test -->

      <div class="mb-4">

        <label for="titreTest" class="block text-sm font-medium text-gray-700">Titre du Test (pour l'utilisateur)</label>

        <input type="text" id="titreTest" name="titreTest" required class="mt-1 block w-full px-3 py-2 bg-white border border-gray-300 rounded-md shadow-sm focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm">

      </div>

      <!-- Type de Test -->

      <div class="mb-4">

        <label for="typeTest" class="block text-sm font-medium text-gray-700">Type de Test</label>

        <select id="typeTest" name="typeTest" required class="mt-1 block w-full pl-3 pr-10 py-2 text-base border-gray-300 focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm rounded-md">

          <option value="">Chargement...</option>

        </select>

      </div>

<!-- Blocs Méta Dynamiques -->

<div class="mb-4">

<label class="block text-sm font-medium text-gray-700">Questions Standards (Blocs Méta)</label>

<div id="meta-blocks-container" class="mt-2 p-3 bg-white border border-gray-300 rounded-md space-y-2">

<p class="text-xs text-gray-500">Chargement des options...</p>

</div>

</div>

      <!-- Informations Dynamiques sur les Questions -->

      <div id="questions-info" class="mb-4 p-3 bg-indigo-50 border border-indigo-200 rounded-md hidden">

        <p class="text-sm text-indigo-700">Nombre de questions disponibles : <span id="question-count" class="font-bold">...</span></p>

      </div>

      <!-- Nombre de Questions à Utiliser -->

      <div class="mb-4">

        <label for="nbQuestions" class="block text-sm font-medium text-gray-700">Nombre de questions à utiliser</label>

        <input type="number" id="nbQuestions" name="nbQuestions" required class="mt-1 block w-full px-3 py-2 bg-white border border-gray-300 rounded-md shadow-sm focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm">

      </div>

      <hr class="my-6">

      <!-- Sections Email -->

<div class="mb-2"><label class="flex items-center"><input type="checkbox" id="sendEmailRepondant" class="h-4 w-4 text-indigo-600 border-gray-300 rounded focus:ring-indigo-500"><span class="ml-2 text-sm text-gray-700">Envoyer un email au répondant ?</span></label></div>

<div id="detailsRepondant" class="pl-6 border-l-2 border-gray-200 ml-2 mb-4 hidden-section"></div>

<div class="mb-2"><label class="flex items-center"><input type="checkbox" id="sendEmailPatron" class="h-4 w-4 text-indigo-600 border-gray-300 rounded focus:ring-indigo-500"><span class="ml-2 text-sm text-gray-700">Envoyer un email au patron ?</span></label></div>

<div id="detailsPatron" class="pl-6 border-l-2 border-gray-200 ml-2 mb-4 hidden-section"></div>

<div class="mb-2"><label class="flex items-center"><input type="checkbox" id="sendEmailFormateur" class="h-4 w-4 text-indigo-600 border-gray-300 rounded focus:ring-indigo-500"><span class="ml-2 text-sm text-gray-700">Envoyer un email au formateur ?</span></label></div>

<div id="detailsFormateur" class="pl-6 border-l-2 border-gray-200 ml-2 mb-4 hidden-section"></div>

      <hr class="my-6">

      <div class="mb-4"><label for="emailDev" class="block text-sm font-medium text-gray-700">Email du Développeur (notifications)</label><input type="email" id="emailDev" name="emailDev" placeholder="Défaut : chanenam@gmail.com" class="mt-1 block w-full px-3 py-2 bg-white border border-gray-300 rounded-md shadow-sm focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"></div>

      <button id="submit-button" class="w-full bg-indigo-600 hover:bg-indigo-700 text-white font-bold py-2 px-4 rounded-md focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500 disabled:bg-gray-400">Créer la Configuration</button>

    </div>

    <div id="feedback-screen" class="hidden text-center py-10"><p id="feedback-message" class="text-lg text-gray-700"></p></div>

    <script>

      // --- INITIALISATION & GESTION DYNAMIQUE ---

      document.addEventListener('DOMContentLoaded', function() {

        google.script.run.withSuccessHandler(populateInitialData).getInitialData();

setupEventListeners();

      });

      function populateInitialData(data) {

        const typeTestSelect = document.getElementById('typeTest');

        typeTestSelect.innerHTML = '<option value="">-- Choisissez un type --</option>';

        data.typesDeTest.forEach(type => {

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

          option.value = type; option.textContent = type;

          typeTestSelect.appendChild(option);

        });

const metaContainer = document.getElementById('meta-blocks-container');

if (data.availableMetaBlocks && data.availableMetaBlocks.length > 0) {

metaContainer.innerHTML = '';

data.availableMetaBlocks.forEach(block => {

const isChecked = (block.id === 'META\_EMAIL' || block.id === 'META\_NOM\_COMPLET');

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

label.className = 'flex items-center text-sm text-gray-800';

label.innerHTML = `

<input type="checkbox" value="${block.id}" class="meta-block-checkbox h-4 w-4 text-indigo-600 border-gray-300 rounded focus:ring-indigo-500" ${isChecked ? 'checked' : ''}>

<span class="ml-2">${block.title}</span>

`;

metaContainer.appendChild(label);

});

} else {

metaContainer.innerHTML = '<p class="text-xs text-gray-500">Aucun bloc méta trouvé dans la BDD.</p>';

}

        populateEmailOptions('detailsRepondant', 'Repondant', data.options);

        populateEmailOptions('detailsPatron', 'Patron', data.options);

        populateEmailOptions('detailsFormateur', 'Formateur', data.options);

      }

function setupEventListeners() {

document.getElementById('typeTest').addEventListener('change', function() {

const selectedType = this.value;

const infoDiv = document.getElementById('questions-info');

const countSpan = document.getElementById('question-count');

if (selectedType) {

infoDiv.classList.remove('hidden');

countSpan.textContent = 'Chargement...';

google.script.run.withSuccessHandler(updateQuestionCount).getQuestionCountForTestType(selectedType);

} else {

infoDiv.classList.add('hidden');

}

});

setupEmailSectionToggle('sendEmailRepondant', 'detailsRepondant');

setupEmailSectionToggle('sendEmailPatron', 'detailsPatron');

setupEmailSectionToggle('sendEmailFormateur', 'detailsFormateur');

document.getElementById('submit-button').addEventListener('click', submitForm);

}

      function submitForm(e) {

        e.preventDefault();

        if (!document.getElementById('titreTest').value || !document.getElementById('typeTest').value || !document.getElementById('nbQuestions').value) {

            alert('Veuillez remplir tous les champs obligatoires (Titre, Type, Nombre de questions).');

            return;

        }

const selectedMetaBlocks = [];

document.querySelectorAll('.meta-block-checkbox:checked').forEach(checkbox => {

selectedMetaBlocks.push(checkbox.value);

});

        const formObject = {

          titre: document.getElementById('titreTest').value,

          type: document.getElementById('typeTest').value,

blocsMeta: selectedMetaBlocks,

          nbQuestions: document.getElementById('nbQuestions').value,

          repondantActif: document.getElementById('sendEmailRepondant').checked,

          repondantQuand: document.getElementById('sendEmailRepondant').checked ? document.getElementById('quandRepondant').value : '',

          repondantContenu: document.getElementById('sendEmailRepondant').checked ? document.getElementById('contenuRepondant').value : '',

          patronActif: document.getElementById('sendEmailPatron').checked,

          patronEmail: document.getElementById('sendEmailPatron').checked ? document.getElementById('emailPatron').value : '',

          patronQuand: document.getElementById('sendEmailPatron').checked ? document.getElementById('quandPatron').value : '',

          patronContenu: document.getElementById('sendEmailPatron').checked ? document.getElementById('contenuPatron').value : '',

          formateurActif: document.getElementById('sendEmailFormateur').checked,

          formateurEmail: document.getElementById('sendEmailFormateur').checked ? document.getElementById('emailFormateur').value : '',

          formateurQuand: document.getElementById('sendEmailFormateur').checked ? document.getElementById('quandFormateur').value : '',

          formateurContenu: document.getElementById('sendEmailFormateur').checked ? document.getElementById('contenuFormateur').value : '',

          devEmail: document.getElementById('emailDev').value

        };

        showFeedbackScreen('Enregistrement en cours...');

        document.getElementById('submit-button').disabled = true;

        google.script.run.withSuccessHandler(onSaveSuccess).withFailureHandler(onSaveFailure).processNewTestConfiguration(formObject);

      }

// --- FONCTIONS UTILITAIRES ---

function populateEmailOptions(containerId, prefix, options) {

const container = document.getElementById(containerId);

let html = '';

if (prefix === 'Patron' || prefix === 'Formateur') {

html += `<div class="mt-2 mb-3"><label for="email${prefix}" class="block text-sm font-medium text-gray-600">Adresse e-mail du ${prefix}</label><input type="email" id="email${prefix}" required class="mt-1 block w-full px-2 py-1 bg-white border border-gray-300 rounded-md shadow-sm focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"></div>`;

}

html += `<div class="mt-2 mb-3"><label for="quand${prefix}" class="block text-sm font-medium text-gray-600">Quand envoyer l'email ?</label><select id="quand${prefix}" class="mt-1 block w-full pl-2 pr-8 py-1.5 text-base border-gray-300 focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm rounded-md">${options[`${prefix}\_Quand`].map(opt => `<option value="${opt}">${opt}</option>`).join('')}</select></div>`;

html += `<div class="mt-2 mb-3"><label for="contenu${prefix}" class="block text-sm font-medium text-gray-600">Quel niveau de contenu ?</label><select id="contenu${prefix}" class="mt-1 block w-full pl-2 pr-8 py-1.5 text-base border-gray-300 focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm rounded-md">${options[`${prefix}\_Contenu`].map(opt => `<option value="${opt}">${opt}</option>`).join('')}</select></div>`;

container.innerHTML = html;

}

function updateQuestionCount(count) {

document.getElementById('question-count').textContent = count;

}

function setupEmailSectionToggle(checkboxId, sectionId) {

document.getElementById(checkboxId).addEventListener('change', function() {

const section = document.getElementById(sectionId);

if (this.checked) {

section.classList.remove('hidden-section');

section.classList.add('visible-section');

} else {

section.classList.remove('visible-section');

section.classList.add('hidden-section');

}

});

}

function onSaveSuccess(message) {

showFeedbackScreen(message);

setTimeout(function() {

google.script.host.close();

}, 2000);

}

function onSaveFailure(error) {

showFeedbackScreen('Erreur : ' + error.message);

document.getElementById('submit-button').disabled = false;

}

function showFeedbackScreen(message) {

document.getElementById('main-form').classList.add('hidden');

const feedbackScreen = document.getElementById('feedback-screen');

document.getElementById('feedback-message').textContent = message;

feedbackScreen.classList.remove('hidden');

}

    </script>

  </body>

</html>

// --- Fin du fichier: FormulaireUI.html ---

// --- Début du fichier: Menu.js ---

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

// == FICHIER : menu.gs (Version Corrigée)

// == VERSION : 3.2 (Correction du transfert de l'ID Gabarit)

// == RÔLE : Logique côté serveur pour l'application web de configuration.

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

const ID\_FEUILLE\_CONFIG = "1kLBqIHZWbHrb4SsoSQcyVsLOmqKHkhSA4FttM5hZtDQ";

// --- SECTION 1 : INTERFACE UTILISATEUR ---

function onOpen() {

SpreadsheetApp.getUi()

.createMenu('🚀 Actions Usine')

.addItem('Configurer un nouveau test...', 'showConfigurationSidebar')

.addToUi();

}

function showConfigurationSidebar() {

const html = HtmlService.createHtmlOutputFromFile('FormulaireUI')

.setTitle('Configuration Usine à Tests')

.setWidth(400);

SpreadsheetApp.getUi().showSidebar(html);

}

// --- SECTION 2 : FONCTIONS APPELÉES PAR L'INTERFACE HTML ---

function getInitialData() {

const ss = SpreadsheetApp.openById(ID\_FEUILLE\_CONFIG);

const optionsSheet = ss.getSheetByName("sys\_Options\_Parametres");

if (!optionsSheet) {

throw new Error("L'onglet 'sys\_Options\_Parametres' est introuvable.");

}

const optionsData = optionsSheet.getDataRange().getValues();

const headers = optionsData.shift();

const optionsMap = {};

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

const options = optionsData.map(row => row[i]).filter(String);

optionsMap[header] = options;

});

// Charger la liste des blocs méta disponibles depuis la BDD

let availableMetaBlocks = [];

try {

const systemIds = getSystemIds();

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

const metaSheet = bdd.getSheetByName('Questions\_META\_FR');

if (metaSheet) {

const metaData = metaSheet.getRange(2, 1, metaSheet.getLastRow() - 1, 3).getValues(); // ID, Type, Titre

availableMetaBlocks = metaData.map(row => ({ id: row[0], title: row[2] })).filter(block => block.id && block.title);

}

} catch(e) {

console.error("Impossible de charger les blocs méta depuis la BDD : " + e.message);

}

return {

typesDeTest: optionsMap['Type\_Test'] || [],

availableMetaBlocks: availableMetaBlocks,

options: {

Repondant\_Quand: optionsMap['Repondant\_Quand'] || [],

Repondant\_Contenu: optionsMap['Repondant\_Contenu'] || [],

Patron\_Quand: optionsMap['Patron\_Quand'] || [],

Patron\_Contenu: optionsMap['Patron\_Contenu'] || [],

Formateur\_Quand: optionsMap['Formateur\_Quand'] || [],

Formateur\_Contenu: optionsMap['Formateur\_Contenu'] || []

}

};

}

function getQuestionCountForTestType(typeTest) {

if (!typeTest) return 0;

try {

const systemIds = getSystemIds();

if (systemIds && systemIds.ID\_BDD) {

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

const questionSheet = bdd.getSheets().find(s => s.getName().startsWith('Questions\_' + typeTest));

if (questionSheet) {

return questionSheet.getLastRow() - 1;

}

}

return 0;

} catch (err) {

Logger.log('Erreur lors du calcul du nombre de questions pour ' + typeTest + ': ' + err.message);

return 0;

}

}

// --- SECTION 3 : TRAITEMENT DE LA SOUMISSION ---

function processNewTestConfiguration(formObject) {

try {

const ss = SpreadsheetApp.openById(ID\_FEUILLE\_CONFIG);

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

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

// S'assurer que les nouvelles colonnes existent avant de continuer

let headers = paramsSheet.getRange(1, 1, 1, paramsSheet.getLastColumn()).getValues()[0];

const requiredHeaders = ['Blocs\_Meta\_A\_Inclure', 'ID\_Gabarit\_Email\_Repondant'];

requiredHeaders.forEach(headerName => {

if (headers.indexOf(headerName) === -1) {

paramsSheet.getRange(1, paramsSheet.getLastColumn() + 1).setValue(headerName);

}

});

// Re-lire les en-têtes au cas où ils auraient été modifiés

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

let emailDev = formObject.devEmail;

if (!emailDev || emailDev.trim() === "") { emailDev = "chanenam@gmail.com"; }

const limiteLignes = getQuestionCountForTestType(formObject.type);

const blocsMetaString = formObject.blocsMeta.join(',');

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

// == DÉBUT DE LA CORRECTION : Traduire le "Niveau de contenu" en ID Gabarit

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

let idGabaritRepondant = ''; // Valeur par défaut : vide

if (formObject.repondantContenu === 'Niveau 1 : Simple') {

idGabaritRepondant = 'RESULTATS\_N1';

} else if (formObject.repondantContenu === 'Niveau 2 : Détaillé') {

idGabaritRepondant = 'RESULTATS\_N2';

}

// Ajoutez d'autres "else if" si vous créez plus de niveaux/gabarits.

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

// == FIN DE LA CORRECTION

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

const dataRow = {

'Id\_Unique': '',

'Titre\_Formulaire\_Utilisateur': formObject.titre,

'Nom\_Fichier\_Complet': '',

'Statut': 'En construction',

'Type\_Test': formObject.type,

'Blocs\_Meta\_A\_Inclure': blocsMetaString,

'ID\_Gabarit\_Email\_Repondant': idGabaritRepondant, // <-- AJOUT de la donnée traduite

'ID\_Dossier\_Cible': '',

'Limite\_Lignes\_A\_Traiter': limiteLignes,

'nbQuestions': formObject.nbQuestions,

'Repondant\_Email\_Actif': formObject.repondantActif ? "Oui" : "Non",

'Repondant\_Quand': formObject.repondantQuand,

'Repondant\_Contenu': formObject.repondantContenu,

'Patron\_Email\_Mode': formObject.patronActif ? "Oui" : "Non",

'Patron\_Quand': formObject.patronQuand,

'Patron\_Contenu': formObject.patronContenu,

'Patron\_Email': formObject.patronEmail,

'Formateur\_Email\_Actif': formObject.formateurActif ? "Oui" : "Non",

'Formateur\_Quand': formObject.formateurQuand,

'Formateur\_Contenu': formObject.formateurContenu,

'Formateur\_Email': formObject.formateurEmail,

'Developpeur\_Email': emailDev,

'ID\_Formulaire\_Cible': '',

'ID\_Sheet\_Cible': ''

};

const nouvelleLigne = headers.map(header => dataRow[header] !== undefined ? dataRow[header] : '');

paramsSheet.appendRow(nouvelleLigne);

return "Configuration enregistrée avec succès !";

} catch (e) {

Logger.log("ERREUR lors de la sauvegarde de la configuration: " + e.toString());

throw new Error("Une erreur interne est survenue lors de la sauvegarde. " + e.message);

}

}

// --- SECTION 4 : FONCTIONS UTILITAIRES ---

function getSystemIds() {

const configSS = SpreadsheetApp.openById(ID\_FEUILLE\_CONFIG);

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;

}

// --- Fin du fichier: Menu.js ---

// Fichier généré le 21/07/2025 à 13:01:09

// --- Début du fichier: Code.js ---

/\*\*

 \* @OnlyCurrentDoc

 \* Crée un menu personnalisé dans l'interface utilisateur de la feuille de calcul

 \* pour lancer les fonctions utilitaires.

 \*/

function onOpen() {

  SpreadsheetApp.getUi()

      .createMenu('⚙️ Utilitaires BDD')

      .addItem('Lister les fichiers d\'un dossier Drive', 'listFilesFromFolder')

      .addToUi();

}

/\*\*

 \* Demande à l'utilisateur l'ID d'un dossier Drive, puis liste tous les fichiers

 \* de ce dossier (et optionnellement des sous-dossiers) à la suite des données.

 \*/

function listFilesFromFolder() {

  const ui = SpreadsheetApp.getUi();

  // 1. Demander l'ID du dossier à l'utilisateur

  const result = ui.prompt(

      'Lister les Fichiers Drive',

      'Veuillez coller l\'ID du dossier Google Drive contenant vos rapports :',

      ui.ButtonSet.OK\_CANCEL);

  if (result.getSelectedButton() !== ui.Button.OK || !result.getResponseText()) {

    return;

  }

  const folderId = result.getResponseText().trim();

// NOUVEAU : Demander si l'on doit inclure les sous-dossiers

const recursiveSearchResponse = ui.alert(

'Recherche approfondie',

'Voulez-vous inclure les fichiers des sous-dossiers ?',

ui.ButtonSet.YES\_NO);

const shouldRecurse = (recursiveSearchResponse === ui.Button.YES);

  try {

const folder = DriveApp.getFolderById(folderId);

const filesToAdd = [];

// NOUVEAU : Lancer la recherche simple ou récursive en fonction de la réponse

if (shouldRecurse) {

// Lancer la recherche récursive

getFilesRecursive(folder, filesToAdd);

} else {

// Lancer la recherche simple (uniquement le dossier racine)

const files = folder.getFiles();

while (files.hasNext()) {

const file = files.next();

filesToAdd.push([file.getName(), file.getId()]);

}

}

    if (filesToAdd.length === 0) {

      ui.alert('Information', `Aucun fichier n'a été trouvé dans le dossier "${folder.getName()}" (et ses sous-dossiers, si l'option était choisie).`, ui.ButtonSet.OK);

      return;

    }

    // 3. Écrire les résultats dans la feuille de calcul

    const ss = SpreadsheetApp.getActiveSpreadsheet();

    let outputSheet = ss.getSheetByName('Liste\_Fichiers\_Drive');

    if (!outputSheet) {

      outputSheet = ss.insertSheet('Liste\_Fichiers\_Drive', 0);

    }

const lastRow = outputSheet.getLastRow();

let startRow;

if (lastRow === 0) {

outputSheet.getRange(1, 1, 1, 2).setValues([['Nom du Fichier', 'ID du Fichier']]);

startRow = 2;

} else {

startRow = lastRow + 1;

}

    outputSheet.getRange(startRow, 1, filesToAdd.length, 2).setValues(filesToAdd);

outputSheet.autoResizeColumns(1, 2);

outputSheet.activate();

    ui.alert('Opération terminée', `${filesToAdd.length} nouveau(x) fichier(s) ont été ajoutés dans l'onglet "Liste\_Fichiers\_Drive".`, ui.ButtonSet.OK);

  } catch (e) {

    Logger.log(e.toString());

    ui.alert('Erreur', 'Impossible d\'accéder au dossier. Veuillez vérifier que l\'ID est correct et que vous avez les droits d\'accès.', ui.ButtonSet.OK);

  }

}

/\*\*

\* Fonction auxiliaire récursive pour lister les fichiers.

\* @param {Folder} folder - Le dossier à parcourir.

\* @param {Array} fileList - Le tableau où ajouter les fichiers trouvés.

\*/

function getFilesRecursive(folder, fileList) {

// Ajouter les fichiers du dossier courant

const files = folder.getFiles();

while (files.hasNext()) {

const file = files.next();

fileList.push([file.getName(), file.getId()]);

}

// Parcourir les sous-dossiers et s'appeler soi-même

const subFolders = folder.getFolders();

while (subFolders.hasNext()) {

const subFolder = subFolders.next();

getFilesRecursive(subFolder, fileList);

}

}

// --- Fin du fichier: Code.js ---

// --- Début du fichier: concat\_scripts\_rep.js ---

const fs = require('fs');

const path = require('path');

// --- CONFIGURATION ---

// Extensions de fichiers à inclure

const scriptExtensions = [

'.js', '.ts', '.jsx', '.tsx', '.py', '.html', '.css', '.scss', '.less',

'.xml', '.php', '.rb', '.java', '.c', '.cpp', '.cs', '.go',

'.sh', '.ps1', '.bat', '.cmd', '.sql', '.vue', '.svelte', '.astro'

];

// Le dossier à scanner

const folderToScan = './'; // À ajuster si besoin

// --- FIN CONFIGURATION ---

// Récupérer le nom du répertoire scanné

const resolvedFolderPath = path.resolve(folderToScan);

const dirName = path.basename(resolvedFolderPath);

// Créer une date au format aammjj

const now = new Date();

const year = String(now.getFullYear()).slice(2); // 2 derniers chiffres de l'année

const month = String(now.getMonth() + 1).padStart(2, '0');

const day = String(now.getDate()).padStart(2, '0');

const dateStamp = `${year}${month}${day}`;

// Nom du fichier de sortie

const outputFileName = `${dirName}\_Script\_${dateStamp}.txt`;

// Créer une chaîne avec date et heure en français

const dateTimeString = `// Fichier généré le ${now.toLocaleDateString('fr-FR')} à ${now.toLocaleTimeString('fr-FR')}\n\n`;

// Initialiser la variable qui accumulera le contenu

let allContent = dateTimeString;

// Fonction récursive

function readFilesRecursively(directory) {

fs.readdirSync(directory).forEach(file => {

const absolutePath = path.join(directory, file);

if (fs.statSync(absolutePath).isDirectory()) {

readFilesRecursively(absolutePath);

} else {

const fileExtension = path.extname(file).toLowerCase();

if (scriptExtensions.includes(fileExtension)) {

allContent += `// --- Début du fichier: ${absolutePath} ---\n`;

allContent += fs.readFileSync(absolutePath, 'utf8');

allContent += `\n// --- Fin du fichier: ${absolutePath} ---\n\n`;

}

}

});

}

// Supprimer le fichier de sortie existant

if (fs.existsSync(outputFileName)) {

fs.unlinkSync(outputFileName);

console.log(`Ancien fichier '${outputFileName}' supprimé.`);

}

// Lancer le traitement

try {

readFilesRecursively(folderToScan);

fs.writeFileSync(outputFileName, allContent, 'utf8');

console.log(`Succès : Tous les scripts ont été exportés dans '${outputFileName}'`);

} catch (error) {

console.error(`Erreur lors de la lecture ou de l'écriture des fichiers : ${error.message}`);

}

// --- Fin du fichier: concat\_scripts\_rep.js ---