Absolument. Voici la synthèse complète de notre session, formatée comme demandé pour être sauvegardée et réutilisée.

Markdown

---

Projet: "Usine à Tests"

Date: "2025-07-14"

Version: "4.0"

---

### 1. Résumé des Échanges du Jour

[cite\_start]La session a été consacrée à la validation fonctionnelle de la chaîne V2 et à l'implémentation d'une nouvelle fonctionnalité majeure. [cite: 3]

Les points essentiels abordés sont :

\* \*\*Correction d'Incohérences V2 :\*\* Le premier test de bout en bout a révélé des incohérences critiques. [cite\_start]Le `[MOTEUR]` et le `[TEMPLATE]` ne lisaient pas la même source de vérité pour déterminer le type de question (ancienne colonne vs. "mode" du JSON). [cite: 5, 14] [cite\_start]Ces deux composants ont été synchronisés pour utiliser exclusivement le JSON, renforçant ainsi un principe directeur du projet. [cite: 6, 15]

\* \*\*Amélioration des "Blocs Méta" :\*\* La gestion des questions standards a été entièrement revue. [cite\_start]D'une gestion manuelle, nous sommes passés, sur votre proposition, à une interface de configuration dynamique qui lit le catalogue de questions depuis la `[BDD]` et les affiche sous forme de cases à cocher. [cite: 9, 11, 12]

\* \*\*Nouvelle Fonctionnalité (Consentement RGPD Dynamique) :\*\* Nous avons conçu et implémenté une solution robuste pour gérer un bloc de consentement RGPD optionnel. Plutôt que de coder le lien en dur, nous avons modifié le `[MOTEUR]` pour qu'il interprète un placeholder (`[LIEN\_FICHIER:RGPD\_FR]`) dans la description de la question, recherche l'ID du document correspondant dans un nouvel onglet (`Liste\_Fichiers\_Drive`) de la `[BDD]`, et génère le lien public dynamiquement.

\* [cite\_start]\*\*Fiabilisation :\*\* La restitution des emails a été rendue plus robuste en cherchant les colonnes Nom/Email par mot-clé plutôt que par en-tête strict.

### 2. État Actuel du Projet

\* \*\*Tâches réalisées :\*\*

\* [cite\_start]Correction de la logique de lecture du type de question dans `[MOTEUR]` (`Moteur V2.js`). [cite: 22]

\* [cite\_start]Correction de la logique de lecture dans `[TEMPLATE]` (`Logique\_Universel.gs`) pour l'aligner sur le `[MOTEUR]`.

\* [cite\_start]Mise à jour majeure de l'interface de configuration (`menu.gs`, `FormulaireUI.html`) pour la sélection dynamique des blocs méta. [cite: 25]

\* Implémentation d'une fonctionnalité de lien dynamique dans `Utils V2.gs` pour les blocs méta (cas du RGPD).

\* Configuration de la `[BDD]` pour supporter cette nouvelle fonctionnalité (ajout de l'onglet `Liste\_Fichiers\_Drive` et du bloc `META\_RGPD\_CONSENT`).

\* \*\*Tâches en cours :\*\*

\* [cite\_start]Génération de ce document de synthèse. [cite: 28]

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

\* [cite\_start]\*\*Action immédiate :\*\* Lancer le test de bout en bout que nous avons configuré en fin de session pour valider le bon fonctionnement de la nouvelle fonctionnalité de consentement RGPD. [cite: 30]

### 3. Contexte Historique du Projet

[cite\_start]L'objectif de la session était de faire passer le projet de sa phase de "compatibilité théorique" V2, issue de la migration des données, à une phase de "robustesse fonctionnelle". [cite: 32] [cite\_start]Les tests ont permis de déceler et corriger les dernières incohérences logiques entre les composants `[MOTEUR]`, `[CONFIG]`, et `[TEMPLATE]`. [cite: 33] [cite\_start]Le projet a maintenant atteint une maturité où la chaîne de production est non seulement fonctionnelle mais aussi flexible et intuitive. [cite: 34]

### 4. Orientations Stratégiques

Les actions du jour ont parfaitement renforcé les principes directeurs de l'architecture V2 :

\* [cite\_start]\*\*JSON comme Source Unique de Vérité :\*\* La correction forçant le `[MOTEUR]` et le `[TEMPLATE]` à lire le `"mode"` du JSON a été une mise en application directe et nécessaire de ce principe. [cite: 37]

\* [cite\_start]\*\*Zéro Convention Implicite :\*\* La transformation de la gestion des blocs méta en une interface à cases à cocher en est l'aboutissement : l'utilisateur déclare explicitement ses choix. [cite: 38, 39]

\* \*\*Le Code plus Intelligent que la Configuration :\*\* La nouvelle fonctionnalité de lien dynamique (`[LIEN\_FICHIER:...]`) est un exemple parfait de ce principe. Le code interprète une instruction de haut niveau, ce qui évite de devoir coder en dur un lien dans la configuration et centralise la gestion des fichiers.

### 5. Architecture Technique & Ressources Clés

#### 5.1. État des Scripts et Fichiers

Voici les versions complètes des scripts modifiés aujourd'hui, prêts à être copiés.

\*\*Projet `[MOTEUR] Usine à Tests V2`\*\*

```javascript

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

// FICHIER : Moteur V2.js

// RÔLE : Fonctions principales de création et d'orchestration des tests.

// VERSION : 4.4 - Logique hybride pour les Blocs Méta (spécial/BDD)

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

function lancerCreationFormulaire() { /\* ... (inchangé) ... \*/ }

function lancerCreationSysteme(rowIndex) {

try {

console.log("Lancement de la création pour la ligne " + rowIndex + "...");

const config = getConfigurationFromRow(rowIndex);

if (config['Statut'].toLowerCase() !== 'en construction') { return null; }

const nomFichierComplet = "[" + config['Type\_Test'] + "] " + config['Titre\_Formulaire\_Utilisateur'];

const systemIds = getSystemIds();

if (!systemIds.ID\_TEMPLATE\_TRAITEMENT\_V2) { throw new Error("Clé 'ID\_TEMPLATE\_TRAITEMENT\_V2' introuvable."); }

let dossierCible;

if (config['ID\_Dossier\_Cible']) {

dossierCible = DriveApp.getFolderById(config['ID\_Dossier\_Cible']);

} else {

if (!systemIds.ID\_DOSSIER\_CIBLE\_GEN) { throw new Error("Clé 'ID\_DOSSIER\_CIBLE\_GEN' introuvable."); }

dossierCible = DriveApp.getFolderById(systemIds.ID\_DOSSIER\_CIBLE\_GEN);

}

const templateFile = DriveApp.getFileById(systemIds.ID\_TEMPLATE\_TRAITEMENT\_V2);

const sheetFile = templateFile.makeCopy(nomFichierComplet, dossierCible);

const form = FormApp.create(nomFichierComplet);

form.setDestination(FormApp.DestinationType.SPREADSHEET, sheetFile.getId());

form.setProgressBar(true);

form.setDescription(config['Titre\_Formulaire\_Utilisateur']);

DriveApp.getFileById(form.getId()).moveTo(dossierCible);

try {

if (!systemIds.ID\_BDD) { throw new Error("Clé 'ID\_BDD' introuvable."); }

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

// --- DÉBUT MODIFICATION V4.4 ---

const blocsMetaConfig = config['Blocs\_Meta\_A\_Inclure'];

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

const metaIds = blocsMetaConfig.split(',').map(id => id.trim().toUpperCase());

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

let metaQuestionsMap = {};

if (metaSheet) {

const metaData = metaSheet.getDataRange().getValues();

const metaHeaders = metaData.shift();

const idCol = metaHeaders.indexOf('ID');

if (idCol !== -1) {

metaQuestionsMap = metaData.reduce((acc, row) => {

acc[row[idCol].toUpperCase()] = row;

return acc;

}, {});

}

}

metaIds.forEach(id => {

switch(id) {

case 'META\_EMAIL':

creerItemFormulaire(form, 'TEXTE\_EMAIL', 'Adresse Email', null, null, null);

break;

case 'META\_NOM\_COMPLET':

creerItemFormulaire(form, 'TEXTE\_COURT', 'Nom Complet', null, null, null);

break;

default:

if (metaQuestionsMap[id]) {

const q\_data = metaQuestionsMap[id];

const [q\_id, q\_type\_old, q\_titre, q\_options, q\_logique, q\_description, q\_params\_json] = q\_data;

let final\_meta\_type = q\_type\_old;

if (q\_params\_json) {

try {

const parsedJson = JSON.parse(q\_params\_json);

if (parsedJson.mode) { final\_meta\_type = parsedJson.mode; }

} catch (e) { console.warn(`Bloc Méta BDD '${id}': JSON invalide.`); }

}

creerItemFormulaire(form, final\_meta\_type, q\_titre, q\_options, q\_description, q\_params\_json);

} else {

console.warn(`Bloc méta ID '${id}' non trouvé.`);

}

break;

}

});

}

// --- FIN MODIFICATION V4.4 ---

const toutesLesFeuillesBDD = bdd.getSheets();

const regexLangues = new RegExp('^Questions\_' + config['Type\_Test'] + '\_([A-Z]{2})$', 'i');

const languesAInclure = [];

toutesLesFeuillesBDD.forEach(feuille => {

const match = feuille.getName().match(regexLangues);

if (match && match[1]) { languesAInclure.push({ code: match[1].toUpperCase(), nomComplet: getLangueFullName(match[1]), feuille: feuille }); }

});

if (languesAInclure.length === 0) { throw new Error("Aucune feuille de questions trouvée pour le type '" + config['Type\_Test'] + "'.");

}

const itemLangue = form.addMultipleChoiceItem().setTitle("Langue / Language").setRequired(true);

const choices = [];

languesAInclure.forEach(langue => {

const page = form.addPageBreakItem().setTitle("Questions (" + langue.nomComplet + ")");

choices.push(itemLangue.createChoice(langue.nomComplet, page));

let nbQuestionsAUtiliser = langue.feuille.getLastRow() - 1;

if (config['nbQuestions'] && config['nbQuestions'] > 0) { nbQuestionsAUtiliser = Math.min(config['nbQuestions'], nbQuestionsAUtiliser); }

if (nbQuestionsAUtiliser <= 0) return;

const questionsData = langue.feuille.getRange(2, 1, nbQuestionsAUtiliser, 7).getValues();

questionsData.forEach((q\_data, index) => {

const [id, type\_old, titre, options, logique, description, params\_json] = q\_data;

let final\_type = type\_old;

if (params\_json) {

try {

const parsedJson = JSON.parse(params\_json);

if (parsedJson.mode) { final\_type = parsedJson.mode; }

} catch (e) { console.warn(`Ligne ${index + 2}: JSON invalide.`); }

}

creerItemFormulaire(form, final\_type, id + ': ' + titre, options, description, params\_json);

if (index === questionsData.length - 1) { page.setGoToPage(FormApp.PageNavigationType.SUBMIT); }

});

});

itemLangue.setChoices(choices);

} catch (e) { throw new Error("Génération des questions impossible. Erreur : " + e.message); }

return { formObject: form, formFile: formFile, sheetFile: sheetFile, nomFichierComplet: nomFichierComplet };

} catch(e) {

SpreadsheetApp.getUi().alert("Erreur ligne " + rowIndex + ": " + e.message);

throw e;

}

}

function orchestrateurCreationAutomatique() { /\* ... (inchangé) ... \*/ }

function lancerDeploiementComplet(rowIndex) { /\* ... (inchangé) ... \*/ }

JavaScript

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

// FICHIER : Utils V2.gs (Projet MOTEUR)

// RÔLE : Fonctions utilitaires pour le Moteur.

// VERSION : 2.6 - Recherche dynamique de liens & fiabilisation du texte d'aide

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

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

function getSystemIds() { /\* ... (inchangé) ... \*/ }

function getConfigurationFromRow(rowIndex) { /\* ... (inchangé) ... \*/ }

function getLangueFullName(code) { /\* ... (inchangé) ... \*/ }

function creerItemFormulaire(form, type, titre, optionsString, description, paramsJSONString) {

// --- DÉBUT MODIFICATION V2.6 ---

let finalDescription = description;

const placeholderRegex = /\[LIEN\_FICHIER:(.\*?)\]/;

const match = description ? description.match(placeholderRegex) : null;

if (match && match[1]) {

const nomFichier = match[1].trim();

try {

const systemIds = getSystemIds();

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

const listeFichiersSheet = bdd.getSheetByName('Liste\_Fichiers\_Drive');

if (listeFichiersSheet) {

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

const fileRow = data.find(row => row[0].toString().trim() === nomFichier);

if (fileRow && fileRow[1]) {

const fileId = fileRow[1].toString().trim();

const fileUrl = `https://drive.google.com/file/d/${fileId}/view`;

finalDescription = description.replace(placeholderRegex, fileUrl);

} else {

finalDescription = description.replace(placeholderRegex, `[ERREUR: Fichier '${nomFichier}' introuvable dans la BDD]`);

}

} else {

finalDescription = description.replace(placeholderRegex, `[ERREUR: Onglet 'Liste\_Fichiers\_Drive' introuvable]`);

}

} catch (e) {

Logger.log("Erreur lors de la recherche du lien de fichier : " + e.message);

finalDescription = description.replace(placeholderRegex, `[ERREUR SCRIPT: ${e.message}]`);

}

}

// --- FIN MODIFICATION V2.6 ---

let params = null;

let choices = [];

let item;

if (paramsJSONString && typeof paramsJSONString === 'string' && paramsJSONString.trim().startsWith('{')) {

try {

params = JSON.parse(paramsJSONString);

} catch (e) {

item = form.addParagraphTextItem().setTitle("[Erreur V2: JSON invalide] " + titre);

if(finalDescription) item.setHelpText(finalDescription);

return;

}

}

if (params && params.options && Array.isArray(params.options) && params.options.length > 0) {

choices = params.options.map(opt => (typeof opt === 'object' && opt !== null) ? opt.libelle : opt);

} else if (optionsString) {

choices = optionsString.split(';').map(String);

}

const formItemType = type ? type.toUpperCase() : '';

if (formItemType.startsWith('QRM')) {

if (choices.length > 0) {

item = form.addCheckboxItem().setTitle(titre).setChoiceValues(choices).setRequired(true);

} else {

item = form.addParagraphTextItem().setTitle("[Erreur QRM: Options manquantes] " + titre);

}

} else if (formItemType.startsWith('QCU')) {

if (choices.length > 0) {

item = form.addMultipleChoiceItem().setTitle(titre).setChoiceValues(choices).setRequired(true);

} else {

item = form.addParagraphTextItem().setTitle("[Erreur QCU: Options manquantes] " + titre);

}

} else if (formItemType === 'ECHELLE') {

const libelles = description ? description.split(';') : [];

if (optionsString && optionsString.split(';').length >= 2) {

const bounds = optionsString.split(';').map(Number);

const scaleItem = form.addScaleItem().setTitle(titre).setBounds(bounds[0], bounds[bounds.length - 1]).setRequired(true);

if (libelles.length === 2) {

scaleItem.setLabels(libelles[0], libelles[1]);

}

item = scaleItem;

} else {

item = form.addParagraphTextItem().setTitle("[Erreur ECHELLE: Bornes manquantes] " + titre);

}

} else if (formItemType === 'TEXTE\_EMAIL') {

const textItem = form.addTextItem().setTitle(titre).setRequired(true);

const emailValidation = FormApp.createTextValidation()

.setHelpText("Veuillez entrer une adresse e-mail valide.")

.requireTextIsEmail()

.build();

item = textItem.setValidation(emailValidation);

} else if (formItemType === 'TEXTE\_COURT') {

item = form.addTextItem().setTitle(titre).setRequired(true);

} else {

item = form.addParagraphTextItem().setTitle("[Type Inconnu: " + type + "] " + titre);

}

if (finalDescription && item && typeof item.setHelpText === 'function') {

if(formItemType !== 'ECHELLE') {

item.setHelpText(finalDescription);

}

}

}

**Projet [CONFIG] Usine à Tests**

JavaScript

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

// FICHIER : menu.gs (Version Harmonisée)

// VERSION : 3.1 - Logique dynamique pour Blocs Méta

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

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

function onOpen() { /\* ... (inchangé) ... \*/ }

function showConfigurationSidebar() { /\* ... (inchangé) ... \*/ }

function getInitialData() {

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

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

if (!optionsSheet) { throw new Error("Onglet 'sys\_Options\_Parametres' 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;

});

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();

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 : " + e.message); }

return {

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

availableMetaBlocks: availableMetaBlocks,

options: { /\* ... (inchangé) ... \*/ }

};

}

function getQuestionCountForTestType(typeTest) { /\* ... (inchangé) ... \*/ }

function processNewTestConfiguration(formObject) {

try {

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

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

const newHeaderName = 'Blocs\_Meta\_A\_Inclure';

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

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

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

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

}

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

const dataRow = {

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

'Statut': 'En construction',

'Type\_Test': formObject.type,

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

'nbQuestions': formObject.nbQuestions,

/\* ... (autres champs inchangés) ... \*/

};

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

paramsSheet.appendRow(nouvelleLigne);

return "Configuration enregistrée avec succès !";

} catch (e) { throw new Error("Sauvegarde impossible. " + e.message); }

}

function getSystemIds() { /\* ... (inchangé) ... \*/ }

HTML

<!DOCTYPE html>

<html>

<head>

<base target="\_top">

<script src="[https://cdn.tailwindcss.com](https://cdn.tailwindcss.com)"></script>

</head>

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

<div id="main-form">

<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 rounded-md space-y-2"></div>

</div>

</div>

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

<script>

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

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

});

function populateInitialData(data) {

/\* ... (remplissage Type de Test inchangé) ... \*/

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

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

let content = '';

data.availableMetaBlocks.forEach(block => {

content += `<div><input type="checkbox" id="${block.id}" name="blocsMeta" value="${block.id}" class="h-4 w-4 rounded border-gray-300 text-indigo-600 focus:ring-indigo-500"><label for="${block.id}" class="ml-2 text-sm text-gray-900">${block.title}</label></div>`;

});

metaContainer.innerHTML = content;

} else {

metaContainer.innerHTML = '<p class="text-sm text-gray-500">Aucun bloc méta disponible.</p>';

}

}

</script>

</body>

</html>

**5.2. Architecture Détaillée et Traçabilité**

* **[CONFIG] Usine à Tests (Google Sheet)**: Panneau de contrôle de l'usine.
  + menu.gs (**MODIFIÉ**): Gère les menus et la logique d'ouverture du formulaire de configuration. Modifié pour appeler

getInitialData qui charge maintenant dynamiquement les blocs méta.

* + FormulaireUI.html (**MODIFIÉ**): L'interface utilisateur (HTML/CSS/JS) pour configurer un nouveau test. Modifiée pour créer dynamiquement des cases à cocher pour chaque bloc méta trouvé.
* **[MOTEUR] Usine à Tests (Apps Script)**: Le cœur qui construit les tests.
  + Moteur V2.js (**MODIFIÉ**): Contient la logique d'orchestration principale. Modifié pour lire la nouvelle colonne

Blocs\_Meta\_A\_Inclure et appeler creerItemFormulaire pour chaque ID.

* + Utils V2.js (**MODIFIÉ**): Fonctions utilitaires, notamment creerItemFormulaire. Modifiée pour ajouter la logique de remplacement du placeholder [LIEN\_FICHIER:...] et fiabiliser l'ajout du texte d'aide.
* **[BDD] Tests & Profils (Google Sheet)**: La base de données centrale.
  + Liste\_Fichiers\_Drive (**AJOUTÉ**): Nouvel onglet pour mapper des noms de fichiers à leurs ID Drive.
  + Questions\_META\_FR (**MODIFIÉ**): L'onglet a été modifié pour inclure le nouveau bloc META\_RGPD\_CONSENT.
* **[TEMPLATE] Kit de Traitement V2 (Google Sheet)**: Le modèle de feuille de calcul de traitement.
  + Logique\_Universel.gs (**MODIFIÉ** - *Code non fourni*): Mentionné comme ayant été corrigé pour lire le "mode" du JSON.
  + TraitementReponses.gs (**MODIFIÉ** - *Code non fourni*): Mentionné comme ayant été fiabilisé pour la recherche du nom/email.

**5.3. Nomenclature des Variables**

| Nom Précis | Type de Donnée | Origine/Portée | Contenu Attendu & Structure | Exemple Concret |
| --- | --- | --- | --- | --- |
| **config** | Object | lancerCreationSysteme (Moteur V2.js) | Objet JS contenant toutes les valeurs de la ligne de configuration. | { 'Statut': 'En construction', 'Titre\_Formulaire\_Utilisateur': 'Test MBTI', 'Blocs\_Meta\_A\_Inclure': 'META\_NOM\_COMPLET,META\_RGPD\_CONSENT' } |
| **finalDescription** | String | creerItemFormulaire (Utils V2.js) | Le texte de la description après traitement du placeholder. | "Consultez la politique ici : https://drive.google.com/file/d/1gR.../view" |
| **availableMetaBlocks** | Array<Object> | getInitialData (menu.gs) | Tableau d'objets représentant les blocs méta disponibles. | [ { id: 'META\_NOM\_COMPLET', title: 'Nom Complet' }, { id: 'META\_RGPD\_CONSENT', title: 'Politique de confidentialité...' } ] |
| **nomFichier** | String | creerItemFormulaire (Utils V2.js) | Le nom du fichier extrait du placeholder. | "RGPD\_FR" |

Exporter vers Sheets

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

* **Fichier : [CONFIG] Usine à Tests**
  + **Onglet Paramètres Généraux** : ID\_Dossier\_Cible, Titre\_Formulaire\_Utilisateur, Type\_Test, nbQuestions, Statut, Blocs\_Meta\_A\_Inclure.
  + **Onglet sys\_ID\_Fichiers** : Nom\_ID, ID\_Fichier.
* **Fichier : [BDD] Tests & Profils (V2)**
  + **Onglet Questions\_META\_FR** : ID, TypeQuestion, Titre, Options, Logique, Description, Paramètres (JSON).
  + **Onglet Liste\_Fichiers\_Drive** : Nom du Fichier, ID du Fichier.
  + **Onglet Questions\_[Type]\_[Langue]** : Mêmes colonnes que Questions\_META\_FR.

**6. Actions Recommandées / Prochaines Étapes**

* **Action Immédiate :** Procéder au test de la chaîne complète en utilisant la nouvelle interface pour configurer un test incluant le bloc META\_RGPD\_CONSENT. Valider que le lien généré dans le formulaire est correct et fonctionnel.
* **Prochaines Sessions :**
  + Enrichir l'onglet

Questions\_META\_FR avec le catalogue de questions standards qui avait été proposé (Email Manager, etc.).

* + Mettre à jour le "Manuel d'Architecture" pour documenter la nouvelle fonctionnalité [LIEN\_FICHIER:...] et le fonctionnement de l'onglet Liste\_Fichiers\_Drive.
  + Réfléchir à d'autres "instructions intelligentes" qui pourraient être ajoutées au moteur sur le même modèle.