---

Projet: Usine à Tests

Date: 2025-07-14

Version: 3.3

---

### 1. Résumé des Échanges du Jour

La session a été extrêmement productive. Après avoir pris la décision stratégique d'abandonner la V1, nous avons procédé à un audit complet des architectures V1 et V2 en analysant manuellement tous les scripts pertinents (`Moteur`, `Kit`, `Utils`). Cette analyse a clarifié la transition et validé la supériorité de la V2.

Nous avons ensuite mené une session de refonte ciblée du \*\*Moteur V2\*\* pour éliminer la dernière "dette technique" de la V1 : les questions codées en dur. Nous les avons remplacées par un système flexible de \*\*"blocs de questions méta"\*\* configurables depuis la feuille `[CONFIG]`, rendant le moteur 100% pilotable par la configuration.

Enfin, nous avons validé l'ensemble de la chaîne de traitement V2 (configuration, génération, calcul) à travers un "test à blanc" et confirmé que l'architecture est désormais robuste, flexible et cohérente. Une prochaine étape clé a été identifiée : la création d'un outil de migration pour les anciennes questions.

### 2. État Actuel du Projet

\* \*\*Tâches réalisées :\*\*

\* Refonte complète du `Moteur V2.js` (v4.2) pour intégrer la logique des blocs de questions méta.

\* Suppression des dernières conventions implicites (questions en dur) du code du Moteur.

\* Validation théorique de l'architecture V2 de bout en bout, de la configuration à la restitution des résultats.

\* \*\*Tâches en cours :\*\*

\* Génération de ce document de synthèse final.

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

\* Standardiser les en-têtes de toutes les feuilles de questions dans la BDD.

\* Développer l'outil de migration des questions V1 vers le format JSON V2.

### 3. Contexte Historique du Projet

L'audit a révélé une évolution du projet en trois phases distinctes :

\* \*\*Version 0 - Le Prototype Artisanal :\*\* Un système "tout-en-un" où une seule feuille de calcul gérait le test "Couleurs" de A à Z.

\* \*\*Version 1 - L'Usine Spécialisée :\*\* Une évolution majeure séparant l'usine (`[MOTEUR]`) du produit (`[TEMPLATE] Kit`), mais où chaque Kit contenait encore une logique de calcul spécialisée.

\* \*\*Version 2 - L'Usine Universelle (Validée) :\*\* L'architecture actuelle, où la logique de calcul est externalisée dans les données (`JSON`) et où un moteur universel (`Logique\_Universel.gs`) traite tous les types de tests.

### 4. Orientations Stratégiques

Les actions du jour ont permis de concrétiser pleinement les principes directeurs de la V2 :

\* \*\*Zéro Convention Implicite :\*\* Ce principe est maintenant totalement respecté dans le Moteur grâce au remplacement des questions en dur par le système de blocs méta configurables.

\* \*\*Le JSON comme Source Unique de Vérité :\*\* Ce principe est le pilier de la nouvelle architecture, validé par notre analyse du `Logique\_Universel.gs`.

\* \*\*Le Code plus Intelligent que la Configuration :\*\* La nouvelle fonction `creerItemFormulaire` (v2.4) et la logique des blocs méta sont des exemples parfaits de cette approche robuste.

### 5. Architecture Technique & Ressources Clés

#### 5.1. État des Scripts et Fichiers

\* \*\*Fichier Modifié : `Moteur V2.js` (Version 4.2)\*\*

\* Le code complet de la version finale que nous avons validée.

```javascript

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

// FICHIER : Moteur V2.js

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

// VERSION : 4.2 - Remplacement des questions en dur par des blocs méta configurables

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

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("La clé 'ID\_TEMPLATE\_TRAITEMENT\_V2' est introuvable.");

}

let dossierCible;

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

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

} else {

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 {

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

// Injection des blocs méta configurables

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

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

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

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

if (metaSheet) {

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

const metaHeaders = metaData.shift();

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

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

acc[row[idCol]] = row;

return acc;

}, {});

metaIds.forEach(id => {

if (metaQuestionsMap[id]) {

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

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

}

});

}

}

// Injection des questions du test

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));

const nbQuestionsDisponibles = langue.feuille.getLastRow() - 1;

let nbQuestionsAUtiliser = (config['nbQuestions'] && config['nbQuestions'] > 0) ? Math.min(config['nbQuestions'], nbQuestionsDisponibles) : nbQuestionsDisponibles;

if (nbQuestionsAUtiliser <= 0) return;

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

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

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

creerItemFormulaire(form, 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("Impossible de générer les questions. Erreur : " + e.message);

}

return { formObject: form, formFile: DriveApp.getFileById(form.getId()), sheetFile: sheetFile, nomFichierComplet: nomFichierComplet };

} catch(e) {

console.error("ERREUR (ligne " + rowIndex + ") : " + e.toString() + "\n" + e.stack);

throw e;

}

}

// Les autres fonctions (orchestrateurCreationAutomatique, lancerDeploiementComplet, etc.) restent inchangées.

**5.2. Architecture Détaillée et Traçabilité**

* **Projet : [MOTEUR] Usine à Tests V2**
  + **Rôle :** Usine 100% configurable qui génère les Kits de test.
  + **Logique :** La fonction lancerCreationSysteme lit maintenant la colonne Blocs\_Meta\_A\_Inclure pour injecter des questions standards avant de traiter les questions spécifiques au test. Les questions en dur ont été **supprimées**.
* **Projet : [TEMPLATE] Kit de Traitement V2**
  + **Rôle :** Modèle autonome pour chaque test.
  + **Logique :** Le fichier TraitementReponses.gs ne contient plus de logique métier. Il agit comme un simple aiguilleur vers le moteur Logique\_Universel.gs, qui contient toute l'intelligence de calcul. Les fichiers Logique\_\*.js spécialisés sont obsolètes.

**5.3. Nomenclature des Variables**

* **Objet : config**
  + **Nouveau Champ Clé :** config['Blocs\_Meta\_A\_Inclure']
  + **Type :** String
  + **Origine :** Lu depuis la colonne Blocs\_Meta\_A\_Inclure de la feuille Paramètres Généraux.
  + **Contenu Attendu :** Une chaîne de caractères contenant des ID de questions méta, séparés par des virgules.
  + **Exemple Concret :** "META\_NOM\_COMPLET,META\_EMAIL"

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

* **Fichier : [CONFIG] Usine à Tests**
  + **Onglet : Paramètres Généraux**
    - **Nouvelle Colonne Requise :** Blocs\_Meta\_A\_Inclure.
* **Fichier : [BDD] Tests & Profils**
  + **Nouvel Onglet Recommandé :** Questions\_META\_FR (et autres langues si besoin).
    - **Rôle :** Stocker les questions standards réutilisables (Nom, Email, etc.).
  + **En-têtes à Standardiser :** Il a été confirmé que les en-têtes des feuilles de questions doivent être standardisés pour correspondre à ceux attendus par Logique\_Universel.gs : ID, TypeQuestion, Titre, Options, Logique, Description, Paramètres (JSON).

**6. Actions Recommandées / Prochaines Étapes**

* **Action Immédiate :** Standardiser les en-têtes de toutes les feuilles de questions (Questions\_\*) dans la BDD pour correspondre au format V2.
* **Tâche de Développement Prioritaire :**
  + **Créer un outil de migration :** Développer une mini-application ou un script Google Apps Script dédié pour convertir les anciennes questions (MBTI, Couleurs, Ancres) au nouveau format V2. Cet outil devra lire les anciennes colonnes (Options, Logique) et générer automatiquement la structure JSON correspondante dans la colonne Paramètres (JSON).
* **Pistes de Réflexion :**
  + Comment gérer la validation des Paramètres (JSON) pour éviter les erreurs de saisie manuelle ? (Ex: un validateur dans l'interface de configuration).
  + Comment enrichir le système de "blocs méta" pour qu'il puisse gérer des blocs de plusieurs questions (ex: un bloc d'introduction complet) ?