**Manuel d'Architecture : Usine à Tests**

**Version :** 1.0 **Date :** 2025-07-14

**1. Vision et Contexte Historique**

Ce système a été conçu pour automatiser entièrement la création de questionnaires (Google Forms) et de leurs feuilles de calcul de traitement associées ("Kits"). L'objectif est de transformer une simple ligne de configuration en un système de test complet, robuste et flexible.

Le projet a connu trois grandes phases d'évolution:

* **Version 0 - Le Prototype Artisanal :** Un système "tout-en-un" où une seule feuille de calcul gérait un unique test de A à Z.
* **Version 1 - L'Usine Spécialisée :** Une séparation entre l'usine de génération [MOTEUR] et le produit [TEMPLATE] Kit, mais chaque Kit contenait encore une logique de calcul qui lui était propre.
* **Version 2 - L'Usine Universelle (Actuelle) :** L'architecture validée où toute la logique de calcul est externalisée dans les données (format JSON) et traitée par un moteur de calcul universel, rendant l'usine agnostique au type de test qu'elle produit.

**2. Principes Directeurs**

Les actions de développement sur la V2 ont pour but de respecter trois principes fondamentaux:

1. **Zéro Convention Implicite :** Tout paramètre ou toute logique doit être explicitement défini dans la configuration. Le code ne doit rien deviner. Le remplacement des questions en dur par les "blocs méta" est une application directe de ce principe.
2. **Le JSON comme Source Unique de Vérité :** Toute la logique de calcul d'une question (type, profils, scores) est contenue dans un champ JSON dans la base de données. Le code n'est qu'un interpréteur de cette configuration.
3. **Le Code plus Intelligent que la Configuration :** Le code doit être conçu pour gérer des configurations variées de manière robuste, plutôt que de nécessiter une configuration parfaitement formatée pour fonctionner.

**3. Architecture Globale des Composants**

Le système s'articule autour de quatre composants principaux :

* **[CONFIG] Usine à Tests**: Feuille de calcul servant de panneau de contrôle. C'est ici que l'utilisateur décrit les tests à générer via un formulaire dédié.
* **[MOTEUR] Usine à Tests** : Le projet Google Apps Script qui lit une ligne de configuration dans

[CONFIG] et orchestre la construction du Formulaire et du Kit de traitement.

* **[BDD] Tests & Profils**: La base de données centrale. Elle contient toutes les questions, les textes des profils, les traductions et les configurations

JSON.

* **[TEMPLATE] Kit de Traitement V2**: Le modèle de feuille de calcul autonome qui est copié pour chaque nouveau test. Son script interne ne contient plus de logique métier, il se contente d'appeler un moteur de calcul universel (

Logique\_Universel.gs) pour traiter les réponses.

**4. Le Workflow Utilisateur**

Le déploiement d'un nouveau test suit un processus simple en trois étapes:

1. **ÉTAPE 1 : CONFIGURER** : Dans

[CONFIG] Usine à Tests, utiliser le menu 🚀 Actions Usine -> 1. Configurer un nouveau test... pour créer une nouvelle ligne de commande avec le statut "En construction".

1. **ÉTAPE 2 : DÉPLOYER** : Dans

[MOTEUR] Usine à Tests, lancer 🏭 Usine à Tests -> 🚀 Déployer un test de A à Z... et fournir le numéro de la ligne à traiter.

1. **ÉTAPE 3 : ACTIVER** : Ouvrir le "Kit de Traitement" nouvellement généré et cliquer une seule fois sur

⚙️ Actions du Kit -> Activer le traitement des réponses pour mettre en place le déclencheur onFormSubmit.

**5. Le Cœur du Système : La Configuration par les Données (JSON)**

La flexibilité du système repose sur la colonne

Paramètres (JSON) dans les feuilles de questions de la BDD. Le moteur de calcul universel interprète ce

JSON pour savoir comment traiter une réponse. Les modes validés sont :

* **QCU\_CAT (Choix Unique - Catégoriel Pondéré)**
  + **Logique :** La réponse à la question ajoute un score (positif ou négatif) à un profil unique pour toute la question.
  + **Exemple JSON :** {"mode": "QCU\_CAT", "profil": "SERVICE", "options": [{"libelle": "C'est ma priorité absolue", "valeur": 5}, "J'aide volontiers si j'ai le temps", {"libelle": "Je le redirige vers le manuel", "valeur": -1}]}
* **QRM\_CAT (Réponses Multiples - Catégoriel)**
  + **Logique :** L'utilisateur peut cocher plusieurs cases. Chaque option cochée est autonome et ajoute un score à son propre profil.
  + **Exemple JSON :** {"mode": "QRM\_CAT", "options": [{"libelle": "J'aime analyser des données", "profil": "ANALYTIQUE", "valeur": 2}, {"libelle": "Je suis doué pour organiser des événements", "profil": "ORGANISATION", "valeur": 2}]}
* **QCU\_DIRECT (Choix Unique - Affectation Directe)**
  + **Logique :** La valeur du profil n'est pas un score numérique, mais le texte exact de la réponse choisie par l'utilisateur. Le

JSON ne contient pas d'options, celles-ci sont prises depuis la colonne Options de la feuille de questions.

* + **Exemple JSON :** {"mode": "QCU\_DIRECT", "profil": "Trait\_Principal"}

**6. Fonctionnalité Clé : Les Blocs Méta**

Pour respecter le principe de "Zéro Convention Implicite", les questions standards (ex: Nom, Email) ne sont plus codées en dur dans le

Moteur. Elles sont désormais gérées comme des "blocs méta".

* **Principe :** Il s'agit de questions standards stockées dans un onglet dédié de la BDD (ex: Questions\_META\_FR).
* **Déclenchement :** Lors de la configuration d'un test dans [CONFIG], une colonne Blocs\_Meta\_A\_Inclure permet de lister les ID des questions méta à injecter au début du formulaire (ex: "META\_NOM\_COMPLET,META\_EMAIL").

**7. Structure de la Base de Données [BDD] Tests & Profils**

La

BDD est le cerveau du système et doit respecter une structure précise.

* **Questions\_[Type]\_[Langue]** : Stocke les questions d'un test. Les colonnes à standardiser sont :

ID, TypeQuestion, Titre, Options, Logique, Description, Paramètres (JSON).

* **Questions\_META\_[Langue]** : Nouvel onglet recommandé pour stocker les questions standards réutilisables (Nom, Email, etc.) sur le même format que les questions de test.
* **Profils\_[Type]\_[Langue]** : Contient les textes de description pour chaque profil de résultat possible. La première colonne est le code invariant du profil.
* **traductions** : Gère la traduction de tous les éléments statiques de l'interface et des emails (boutons, titres, phrases-types) via un système de clé-valeur.
* **sys\_PiecesJointes** : Définit les fichiers à joindre aux emails de restitution en fonction du contexte (type de test, profil, langue, etc.).

**8. Dépannage (Troubleshooting)**

* **Problème :** Une question de type QRM (cases à cocher) s'affiche avec des boutons radio.
  + **Solution :** Le script Utils V2.gs du Moteur est obsolète (antérieur à v2.4). Il faut le mettre à jour.
* **Problème :** Une question QCU\_DIRECT génère une erreur [Erreur V2: JSON sans options].
  + **Solution :** Le script Utils V2.gs est obsolète (antérieur à v2.3). La mise à jour gère correctement les JSON sans clé "options".
* **Problème :** L'email de résultat mélange les langues.
  + **Solution :** Une clé de traduction est manquante dans l'onglet traductions de la BDD.
* **Problème :** Les pièces jointes ne sont pas envoyées.
  + **Solution :** Vérifier le nom de l'onglet sys\_PiecesJointes, ses en-têtes, la validité de l'ID du fichier et ses autorisations de partage sur Drive.