**Absolument. Voici la synthèse complète de nos échanges et de l'état final du projet, conformément à votre demande et au prompt fourni.**

**Markdown**

**---**

**Projet: Usine à Tests**

**Date: 2025-07-07**

**Version: 8.0 (Production Stable)**

**---**

**### 1. Résumé des Échanges du Jour**

**La session a été consacrée à la finalisation de l'architecture du `[TEMPLATE] Kit de Traitement` après avoir découvert des bugs lors des tests de la chaîne complète.**

**\* \*\*Identification du Problème Initial :\*\* Un premier test a échoué car les scores des profils étaient systématiquement à zéro. Le diagnostic a révélé que la logique de calcul dans `Logique\_Couleurs.gs` n'était plus synchronisée avec la structure des données.**

**\* \*\*Refonte de la Logique de Calcul :\*\* Nous avons redéfini collaborativement une architecture de calcul robuste, basée sur des codes invariants (`Q01`, `R`, `J`, etc.) et une logique de correspondance par index, rendant le calcul totalement indépendant de la langue.**

**\* \*\*Découverte de Bugs en Cascade :\*\* La mise en place de cette nouvelle logique a révélé une série de bugs de dépendance :**

**1. \*\*`getSystemIds is not defined` :\*\* Le script `Logique\_Couleurs.gs` appelait une fonction qui n'existait que dans le projet `MOTEUR`. La solution a été d'ajouter cette fonction aux `Utilities.gs` du Kit pour le rendre autonome.**

**2. \*\*`Cannot read properties of undefined (reading 'Type\_Test')` :\*\* La fonction centrale `processSubmissionData` n'avait pas été mise à jour pour passer tous les paramètres nécessaires (`langue`, `config`) à la nouvelle fonction de calcul.**

**\* \*\*Internationalisation Complète :\*\* Nous avons identifié que le "cadre" textuel des e-mails (salutations, titres) n'était pas traduit. Cela a mené à une dernière refonte majeure :**

**\* Création d'un onglet `traductions` dans la BDD pour centraliser tous les textes.**

**\* Création d'une fonction `loadTraductions` dans `Utilities.gs`.**

**\* Refonte complète de `buildAndSendEmails` pour qu'elle assemble dynamiquement les e-mails à partir des traductions.**

**\* Simplification de `Logique\_Couleurs.gs` pour qu'il ne retourne que des données brutes, séparant ainsi le calcul du formatage.**

**\* \*\*Amélioration Ergonomique :\*\* Nous avons implémenté une nouvelle fonctionnalité permettant de retraiter une réponse dans une langue spécifique, renforçant les capacités d'administration du système.**

**### 2. État Actuel du Projet**

**Le projet a atteint une version stable et fonctionnelle.**

**\* \*\*Tâches réalisées :\*\***

**\* [cite\_start]Finalisation et validation de la chaîne de traitement complète, de la soumission du formulaire à l'envoi de l'e-mail. [cite: 3]**

**\* Mise en place d'une architecture de calcul et de reporting robuste, modulaire et entièrement internationalisée.**

**\* Ajout de la fonctionnalité de retraitement multilingue.**

**\* [cite\_start]Le module `CONFIG` est stable. [cite: 15]**

**\* Le module `[TEMPLATE] Kit de Traitement` est maintenant stable et autonome. [cite: 16]**

**\* \*\*Tâches en cours :\*\* Aucune. Le système est en attente de la prochaine phase d'évolution.**

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

**\* Créer les fichiers de logique pour les autres tests (MBTI, Ancres).**

**\* Aborder les nouvelles améliorations ergonomiques souhaitées. [cite: 21]**

**### 3. Contexte Historique du Projet**

**Partant d'un système fonctionnel, le projet a subi une refonte majeure pour améliorer l'ergonomie (via une Web App) et la robustesse. [cite: 23] Les tests d'intégration récents ont été cruciaux pour révéler des failles de cohérence entre les modules[cite: 24], notamment :**

**\* [cite\_start]Un bug critique où le déclencheur `onFormSubmitTrigger` n'était pas créé. [cite: 5]**

**\* [cite\_start]La décision d'architecture de rendre le Kit de Traitement autonome en le laissant créer son propre déclencheur via un menu dédié. [cite: 6, 7]**

**\* Un problème d'harmonisation où le `MOTEUR` n'utilisait pas les bons en-têtes pour mettre à jour la feuille de configuration après une génération. [cite: 8, 9, 10]**

**### 4. Orientations Stratégiques**

**Les échanges ont renforcé les orientations stratégiques du projet :**

**\* [cite\_start]\*\*Autonomie des Composants :\*\* La décision de faire gérer son propre déclencheur par le Kit a prouvé la valeur de cette approche pour la robustesse du système. [cite: 28]**

**\* \*\*Découplage Logique/Données :\*\* La nouvelle architecture qui sépare les calculs (codes invariants), les données de profil (BDD) et les textes d'interface (`traductions`) est le pilier de la scalabilité et de la maintenabilité du projet.**

**\* [cite\_start]\*\*Validation par le Test :\*\* L'approche itérative de test de bout en bout a été la seule méthode efficace pour découvrir les failles d'intégration profondes. [cite: 27]**

**### 5. Derniers Scripts et Ressources Clés**

**Voici les versions finales et complètes des 3 scripts composant le `[TEMPLATE] Kit de Traitement`, ainsi que la structure de la ressource clé `traductions`.**

**#### Fichier `TraitementReponses.gs`**

**```javascript**

**// =================================================================================**

**// == FICHIER : TraitementReponses.gs (Le Chef d'Orchestre)**

**// == VERSION : 2.3 - Finalisation de l'internationalisation.**

**// =================================================================================**

**function onOpen() {**

**SpreadsheetApp.getUi()**

**.createMenu('⚙️ Actions du Kit')**

**.addItem('Activer le traitement des réponses', 'creerDeclencheur')**

**.addSeparator()**

**.addItem('Retraiter une réponse...', 'retraiterReponse\_UI')**

**.addToUi();**

**}**

**function creerDeclencheur() {**

**const ss = SpreadsheetApp.getActiveSpreadsheet();**

**const declencheurs = ScriptApp.getUserTriggers(ss);**

**declencheurs.forEach(trigger => {**

**if (trigger.getHandlerFunction() === 'onFormSubmitTrigger') {**

**ScriptApp.deleteTrigger(trigger);**

**}**

**});**

**ScriptApp.newTrigger('onFormSubmitTrigger')**

**.forSpreadsheet(ss)**

**.onFormSubmit()**

**.create();**

**SpreadsheetApp.getUi().alert('✅ Le traitement des réponses est maintenant activé pour ce test.');**

**}**

**function retraiterReponse\_UI() {**

**const ui = SpreadsheetApp.getUi();**

**const response = ui.prompt(**

**'Retraitement Multilingue',**

**'Entrez le N° de ligne, suivi du code langue (ex: 2,EN ou 3,FR).\nLaissez la langue vide pour utiliser la langue d\'origine.',**

**ui.ButtonSet.OK\_CANCEL);**

**if (response.getSelectedButton() == ui.Button.OK) {**

**try {**

**const inputText = response.getResponseText();**

**const parts = inputText.split(',').map(p => p.trim());**

**const rowIndex = parseInt(parts[0], 10);**

**const langueCode = parts[1] ? parts[1].toUpperCase() : null;**

**if (!isNaN(rowIndex) && rowIndex > 1) {**

**retraiterReponse(rowIndex, langueCode);**

**ui.alert('Le retraitement de la ligne ' + rowIndex + ' a été lancé. L\'e-mail sera envoyé dans la langue demandée.');**

**} else {**

**ui.alert('Numéro de ligne invalide. Veuillez entrer un nombre supérieur à 1.');**

**}**

**} catch (e) {**

**ui.alert('Format d\'entrée invalide. Assurez-vous d\'utiliser le format "ligne,langue", par exemple "2,EN".');**

**}**

**}**

**}**

**function onFormSubmitTrigger(e) {**

**processSubmissionData(e.namedValues, null);**

**}**

**function retraiterReponse(rowIndex, langueCode) {**

**try {**

**const ss = SpreadsheetApp.getActiveSpreadsheet();**

**const reponsesSheet = ss.getSheets().find(s => s.getName().startsWith("Réponses au formulaire"));**

**if (!reponsesSheet) { throw new Error("Impossible de trouver l'onglet des réponses au formulaire."); }**

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

**const rowData = reponsesSheet.getRange(rowIndex, 1, 1, reponsesSheet.getLastColumn()).getValues()[0];**

**const namedValues = {};**

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

**if (header) { namedValues[header] = [rowData[i]]; }**

**});**

**Logger.log(`Retraitement manuel lancé pour la ligne ${rowIndex} en langue : ${langueCode || 'd\'origine'}`);**

**processSubmissionData(namedValues, langueCode);**

**} catch (err) {**

**Logger.log("ERREUR lors du retraitement de la ligne " + rowIndex + " : " + err.toString());**

**SpreadsheetApp.getUi().alert("Erreur lors du retraitement : " + err.message);**

**}**

**}**

**function processSubmissionData(namedValues, langueOverride) {**

**try {**

**Logger.log("Début du traitement des données.");**

**const config = getTestConfiguration();**

**const reponse = {**

**nomRepondant: namedValues['Nom Complet / Full Name'][0],**

**emailRepondant: namedValues['Adresse Email'][0],**

**toutesLesReponses: namedValues**

**};**

**Logger.log("Données de : " + reponse.nomRepondant + " (" + reponse.emailRepondant + ")");**

**let langueCode;**

**if (langueOverride) {**

**langueCode = langueOverride.toUpperCase();**

**Logger.log("Langue de sortie forcée : " + langueCode);**

**} else {**

**const langueRepondant = namedValues['Langue / Language'][0];**

**const mapLangue = { 'Français': 'FR', 'English': 'EN', 'Español': 'ES', 'Deutsch': 'DE' };**

**langueCode = mapLangue[langueRepondant] || 'FR';**

**}**

**let resultats;**

**switch (config.Type\_Test) {**

**case 'Couleurs':**

**resultats = calculerResultatCouleurs(reponse.toutesLesReponses, langueCode, config);**

**break;**

**case 'MBTI':**

**// resultats = calculerResultatMBTI(reponse.toutesLesReponses, langueCode, config);**

**break;**

**case 'Ancres':**

**// resultats = calculerResultatAncres(reponse.toutesLesReponses, langueCode, config);**

**break;**

**default:**

**Logger.log("Type de test non reconnu ou non géré : " + config.Type\_Test);**

**return;**

**}**

**Logger.log("Résultats calculés pour le test " + config.Type\_Test);**

**buildAndSendEmails(config, reponse, resultats, langueCode);**

**Logger.log("Traitement des données terminé avec succès pour " + reponse.emailRepondant);**

**} catch (err) {**

**Logger.log("ERREUR CRITIQUE dans processSubmissionData : " + err.toString() + "\n" + err.stack);**

**const adminEmail = "chanenam@gmail.com";**

**try {**

**MailApp.sendEmail(adminEmail, "Erreur Critique dans un Kit de Traitement", "Une erreur est survenue dans le script du test " + SpreadsheetApp.getActiveSpreadsheet().getName() + ".\n\nErreur : " + err.message + "\n\nStack Trace:\n" + err.stack);**

**} catch (e) {**

**Logger.log("Impossible d'envoyer l'email de notification d'erreur au développeur.");**

**}**

**}**

**}**

**Fichier Logique\_Couleurs.gs**

**JavaScript**

**// =================================================================================**

**// == FICHIER : Logique\_Couleurs.gs (Le Spécialiste)**

**// == RÔLE : Calcule les résultats et retourne des données BRUTES, non formatées.**

**// =================================================================================**

**function calculerResultatCouleurs(reponses, langue, config) {**

**Logger.log(`Début du calcul par CODE pour le test Couleurs en langue : ${langue}.`);**

**const systemIds = getSystemIds();**

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

**const nomFeuilleQuestions = `Questions\_${config['Type\_Test']}\_${langue}`;**

**const questionsSheet = bdd.getSheetByName(nomFeuilleQuestions);**

**if (!questionsSheet) { throw new Error(`Feuille de questions '${nomFeuilleQuestions}' introuvable.`); }**

**const questionsData = questionsSheet.getDataRange().getValues();**

**const qHeaders = questionsData.shift();**

**const qIdCol = qHeaders.indexOf('ID');**

**const qOptionsCol = qHeaders.indexOf('Options');**

**const qLogiqueCol = qHeaders.indexOf('Logique');**

**const questionsMap = {};**

**questionsData.forEach(row => {**

**const qId = row[qIdCol];**

**if (qId) {**

**questionsMap[qId] = {**

**options: row[qOptionsCol].split(';').map(item => item.trim()),**

**logique: row[qLogiqueCol].split(';').map(item => item.trim())**

**};**

**}**

**});**

**const scores = { 'R': 0, 'J': 0, 'V': 0, 'B': 0 };**

**for (const questionTitre in reponses) {**

**if (!questionTitre.startsWith('Q')) continue;**

**const reponseUtilisateur = reponses[questionTitre][0];**

**const questionId = questionTitre.split(':')[0];**

**const questionInfo = questionsMap[questionId];**

**if (!questionInfo) continue;**

**const reponseIndex = questionInfo.options.indexOf(reponseUtilisateur);**

**if (reponseIndex !== -1) {**

**const codeLogique = questionInfo.logique[reponseIndex];**

**if (scores.hasOwnProperty(codeLogique)) { scores[codeLogique]++; }**

**}**

**}**

**Logger.log("Scores par code calculés : " + JSON.stringify(scores));**

**const profilFinalCode = Object.keys(scores).reduce((a, b) => scores[a] > scores[b] ? a : b);**

**Logger.log("Code du profil dominant déterminé : " + profilFinalCode);**

**const nomFeuilleProfils = `Profils\_Couleurs\_${langue}`;**

**const profilsSheet = bdd.getSheetByName(nomFeuilleProfils);**

**if (!profilsSheet) { throw new Error(`L'onglet de profils '${nomFeuilleProfils}' est introuvable.`); }**

**const profilsData = profilsSheet.getRange(2, 1, profilsSheet.getLastRow() - 1, profilsSheet.getLastColumn()).getValues();**

**const pHeaders = profilsSheet.getRange(1, 1, 1, profilsSheet.getLastColumn()).getValues()[0];**

**const typeCol = pHeaders.indexOf('Type (Couleur)');**

**const nomCompletCol = pHeaders.indexOf('Nom complet');**

**const titreCol = pHeaders.indexOf('Titre du Profil');**

**const descCol = pHeaders.indexOf('Description Complète');**

**const mapCodeToName = {};**

**profilsData.forEach(row => { mapCodeToName[row[typeCol]] = row[nomCompletCol]; });**

**const profilInfo = profilsData.find(row => row[typeCol] === profilFinalCode);**

**const resultats = {**

**profilFinal: profilInfo ? profilInfo[nomCompletCol] : "Inconnu",**

**titreProfil: profilInfo ? profilInfo[titreCol] : "Profil Inconnu",**

**descriptionProfil: profilInfo ? profilInfo[descCol] : "Description non disponible.",**

**scoresData: scores,**

**mapCodeToName: mapCodeToName**

**};**

**Logger.log("Résultats bruts construits : " + JSON.stringify(resultats));**

**return resultats;**

**}**

**Fichier Utilities.gs**

**JavaScript**

**// =================================================================================**

**// == FICHIER : Utilities.gs - VERSION 2.0 (Internationalisé)**

**// =================================================================================**

**const ID\_FEUILLE\_PILOTE = "1kLBqIHZWbHrb4SsoSQcyVsLOmqKHkhSA4FttM5hZtDQ";**

**function getTestConfiguration() {**

**const ss = SpreadsheetApp.getActiveSpreadsheet();**

**const idSheetActuelle = ss.getId();**

**const piloteSheet = SpreadsheetApp.openById(ID\_FEUILLE\_PILOTE);**

**const paramsSheet = piloteSheet.getSheetByName("Paramètres Généraux");**

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

**const data = paramsSheet.getDataRange().getValues();**

**const headers = data.shift();**

**const idSheetColIndex = headers.indexOf('ID\_Sheet\_Cible');**

**if (idSheetColIndex === -1) { throw new Error("La colonne 'ID\_Sheet\_Cible' est introuvable."); }**

**const configRow = data.find(row => row[idSheetColIndex] === idSheetActuelle);**

**if (!configRow) { throw new Error("Impossible de trouver la configuration pour ce test (ID: " + idSheetActuelle + ").");}**

**const configuration = {};**

**headers.forEach((header, index) => {**

**if (header) { configuration[header] = configRow[index]; }**

**});**

**return configuration;**

**}**

**function getSystemIds() {**

**try {**

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

**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 => {**

**const key = row[0];**

**const value = row[1];**

**if (key && value) { ids[key] = value; }**

**});**

**return ids;**

**} catch (e) {**

**Logger.log("Impossible de charger les ID système : " + e.toString());**

**throw new Error("Impossible de charger les ID système. Erreur: " + e.message);**

**}**

**}**

**function loadTraductions(langueCode) {**

**const systemIds = getSystemIds();**

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

**const traductionsSheet = bdd.getSheetByName("traductions");**

**if (!traductionsSheet) throw new Error("L'onglet 'traductions' est introuvable.");**

**const data = traductionsSheet.getDataRange().getValues();**

**const headers = data.shift();**

**const langColIndex = headers.findIndex(h => h.toLowerCase() === langueCode.toLowerCase());**

**if (langColIndex === -1) throw new Error(`La colonne de langue '${langueCode}' est introuvable dans l'onglet 'traductions'.`);**

**const traductions = {};**

**const keyColIndex = 0;**

**data.forEach(row => {**

**const key = row[keyColIndex];**

**const value = row[langColIndex];**

**if (key) {**

**traductions[key] = value;**

**}**

**});**

**Logger.log("Traductions chargées pour la langue " + langueCode);**

**return traductions;**

**}**

**function buildAndSendEmails(config, reponse, resultats, langueCode) {**

**const T = loadTraductions(langueCode);**

**let corpsFinal = "";**

**let sujet = T.SUJET\_EMAIL\_RESULTATS || `Résultats de votre test : ${config.Type\_Test}`;**

**corpsFinal += (T.SALUTATION || "Bonjour {nom\_repondant},").replace('{nom\_repondant}', reponse.nomRepondant) + "\n\n";**

**if (config.Type\_Test === 'Couleurs') {**

**corpsFinal += T.INTRO\_ANALYSE\_COMPLETE\_COULEURS || "Voici l'analyse complète de votre profil de personnalité :\n\n";**

**} else {**

**corpsFinal += T.INTRO\_ANALYSE\_COMPLETE\_GEN || "Voici l'analyse complète de vos résultats :\n\n";**

**}**

**if (config.Type\_Test === 'Couleurs' && resultats.scoresData) {**

**corpsFinal += T.INTRO\_SCORES\_DETAILLES\_COULEURS + "\n";**

**const scoresTries = Object.entries(resultats.scoresData).sort((a, b) => b[1] - a[1]);**

**scoresTries.forEach(([code, score]) => {**

**const nomCouleur = resultats.mapCodeToName[code] || code;**

**corpsFinal += `- ${nomCouleur} : ${score} ${T.SUFFIXE\_POINTS}\n`;**

**});**

**corpsFinal += "\n";**

**corpsFinal += (T.CONCLUSION\_DOMINANTE\_COULEURS || "Votre couleur dominante est le {profilFinal}.").replace('{profilFinal}', resultats.profilFinal) + "\n\n";**

**corpsFinal += (T.INTRO\_SIGNIFICATION\_PROFIL || "Découvrez la signification de votre profil :") + "\n";**

**corpsFinal += resultats.descriptionProfil + "\n\n";**

**}**

**corpsFinal += (T.FORMULE\_POLITESSE || "Bien à vous,") + "\n";**

**const destinataires = [];**

**if (config.Repondant\_Email\_Actif === 'Oui') { destinataires.push({ adresse: reponse.emailRepondant }); }**

**if (config.Patron\_Email\_Mode === 'Oui' && config.Patron\_Email) { destinataires.push({ adresse: config.Patron\_Email }); }**

**if (config.Formateur\_Email\_Actif === 'Oui' && config.Formateur\_Email) { destinataires.push({ adresse: config.Formateur\_Email }); }**

**if (config.Developpeur\_Email) { destinataires.push({ adresse: config.Developpeur\_Email }); }**

**destinataires.forEach(dest => {**

**try {**

**MailApp.sendEmail({ to: dest.adresse, subject: sujet, body: corpsFinal });**

**Logger.log(`Email envoyé avec succès à ${dest.adresse} en ${langueCode}.`);**

**} catch (e) {**

**Logger.log(`ERREUR lors de l'envoi de l'email à ${dest.adresse}: ${e.message}`);**

**}**

**});**

**}**

**Ressource Clé : Structure de l'onglet traductions**

| **Clé** | **fr** | **en** |
| --- | --- | --- |
| **SUJET\_EMAIL\_RESULTATS** | **Résultats de votre test** | **Results of your test** |
| **SALUTATION** | **Bonjour {nom\_repondant},** | **Hello {nom\_repondant},** |
| **INTRO\_ANALYSE\_COMPLETE\_COULEURS** | **Voici votre analyse de profil...** | **Here is your profile analysis...** |
| **INTRO\_SCORES\_DETAILLES\_COULEURS** | **Voici le détail de vos scores :** | **Here is the detail of your scores:** |
| **SUFFIXE\_POINTS** | **points** | **points** |
| **CONCLUSION\_DOMINANTE\_COULEURS** | **Votre couleur dominante est...** | **Your dominant color is...** |
| **INTRO\_SIGNIFICATION\_PROFIL** | **Découvrez votre profil :** | **Discover your profile:** |
| **FORMULE\_POLITESSE** | **Bien à vous,** | **Best regards,** |

**Exporter vers Sheets**

**6. Actions Recommandées / Prochaines Étapes**

* **Actions Immédiates :**
  + **Valider que ce document de synthèse est complet et correct.**
  + **Préparer les données pour les tests MBTI et Ancres dans la BDD en suivant la structure définie.**
* **Questions pour la Prochaine Session :**
  + **Maintenant que la base technique est stable, quelles sont les améliorations ergonomiques prioritaires à aborder ?**
  + **Sommes-nous prêts à développer les scripts de logique Logique\_MBTI.gs et Logique\_Ancres.gs ?**