Absolument. La session d'aujourd'hui a été très productive, marquant une évolution majeure du projet. Voici la synthèse complète et détaillée, prête à être sauvegardée.

Markdown

---

Projet: "Usine à Tests"

Date: "2025-07-16"

Version: "6.0"

---

### 1. Résumé des Échanges du Jour

La session a été consacrée à une évolution majeure de l'architecture : le passage d'un système d'e-mails de restitution statique à un \*\*moteur d'envoi dynamique entièrement piloté par la base de données\*\*.

Les points essentiels abordés sont :

\* \*\*Conception de la nouvelle architecture :\*\* Nous avons validé le principe d'externaliser toute la logique de contenu des e-mails (textes, sujets, niveaux de détail, pièces jointes) dans la base de données `[BDD] Tests & Profils`.

\* \*\*Création des gabarits d'e-mails :\*\*

\* [cite\_start]Nous avons créé la structure d'un nouvel onglet, \*\*`Gabarits\_Emails`\*\*, pour héberger ces modèles. [cite: 5]

\* [cite\_start]Une discussion clé a permis de pivoter d'une approche "spécifique par test" (ex: `MBTI\_N1`) à une approche \*\*"générique et réutilisable"\*\* (ex: `RESULTATS\_N1`), plus robuste et scalable. [cite: 7]

\* \*\*Développement et Débogage :\*\*

\* Le script \*\*`Utilities.gs`\*\* a été profondément remanié pour implémenter cette nouvelle logique.

\* [cite\_start]Un premier test a échoué, mais l'analyse des journaux de `LoggerLib` nous a permis d'identifier et de corriger rapidement un bug lié à la transmission de la langue (`langueCode`) à une sous-fonction. [cite: 17, 19]

\* \*\*Validation et Succès :\*\*

\* Le second test de bout en bout a été un \*\*succès complet\*\*. [cite\_start]L'e-mail a été envoyé correctement, en utilisant le bon gabarit, en injectant les bonnes variables et en incluant les pièces jointes appropriées. [cite: 27]

\* [cite\_start]\*\*Planification du Chantier Suivant :\*\* Nous avons confirmé que le premier chantier étant terminé, la prochaine étape sera de construire l'\*\*interface HTML ergonomique\*\* pour le retraitement des réponses. [cite: 27]

### 2. État Actuel du Projet

\* \*\*Tâches réalisées :\*\*

\* Conception et validation de l'architecture des e-mails dynamiques.

\* Création de la structure de l'onglet `Gabarits\_Emails` dans la `[BDD]`.

\* Mise à jour de la feuille `[CONFIG]` avec les nouvelles colonnes requises (`ID\_Gabarit\_Email\_Repondant`, `Formateur\_Nom`).

\* Réécriture majeure du script `Utilities.gs` (v4.2) pour intégrer la lecture des gabarits, l'injection de variables et la gestion des pièces jointes par niveaux.

\* Correction du bug de `langueCode` et validation de la robustesse de la chaîne de traitement via un test réussi.

\* \*\*Tâches en cours :\*\*

\* Génération de ce document de synthèse.

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

\* \*\*Action immédiate :\*\* Démarrer le "Chantier N°2" en modifiant la fonction `retraiterReponse\_UI` dans `TraitementReponses.gs` pour qu'elle affiche une interface HTML.

\* Concevoir l'interface `RetraitementUI.html` qui permettra de choisir les options d'envoi à la volée.

\* Mettre à jour le "Manuel d'Architecture" pour documenter ces changements majeurs.

### 3. Contexte Historique du Projet

Cette session s'inscrit dans la continuité de la stabilisation de la chaîne V2 et de la mise en place d'outils de débogage robustes. Après avoir solidifié la "mécanique" du projet, nous avons entamé la première grande évolution fonctionnelle : rendre les communications, qui étaient le dernier maillon "rigide" de la chaîne, aussi flexibles et pilotées par les données que le reste du système. Cette étape transforme l'usine d'un simple générateur à un véritable outil de communication personnalisable.

### 4. Orientations Stratégiques

La session d'aujourd'hui est une application directe et quasi parfaite de nos principes directeurs :

\* \*\*Le Code plus Intelligent que la Configuration :\*\* Ce principe a été le moteur de la session. En déplaçant l'intégralité du contenu et de la structure des e-mails dans la `[BDD]`, le code `Utilities.gs` a été transformé en un interpréteur intelligent de la configuration, capable de construire une infinité de types d'e-mails sans une seule modification de son code.

\* \*\*Zéro Convention Implicite :\*\* Notre discussion sur les `ID\_Gabarit` (spécifiques vs. génériques) a été une mise en pratique de ce principe. Nous avons rendu le lien entre un test et son gabarit d'e-mail totalement explicite via la colonne `ID\_Gabarit\_Email\_Repondant`, éliminant toute ambiguïté.

\* \*\*JSON comme Source Unique de Vérité :\*\* Bien que ce ne soit pas du JSON, le principe est étendu aux données en général. La `[BDD]` est plus que jamais la source de vérité unique, non seulement pour les questions, mais aussi pour tout le contenu de restitution.

### 5. Architecture Technique & Ressources Clés

#### 5.1. État des Scripts et Fichiers

Voici la version finale et complète du script `Utilities.gs` que nous avons stabilisé aujourd'hui.

\*\*Projet `[TEMPLATE]V2 Kit de Traitement`\*\*

\*\*Fichier `Utilities.gs` (Version 4.2)\*\*

```javascript

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

// == FICHIER : Utilities.gs - VERSION 4.2 (Correctif de la recherche de PJ)

// == RÔLE : Boîte à outils du Kit de Traitement. Contient la logique de récupération

// == des configurations, des gabarits d'e-mails et de gestion des pièces jointes.

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

// ID de la feuille de calcul centrale qui pilote toute l'usine.

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

/\*\*

\* Récupère la ligne de configuration complète pour le test en cours.

\*/

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;

}

/\*\*

\* Lit l'onglet 'sys\_ID\_Fichiers' de la feuille de configuration centrale.

\*/

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 => {

if (row[0] && row[1]) { ids[row[0]] = row[1]; }

});

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);

}

}

/\*\*

\* NOUVEAU : Récupère une ligne de gabarit d'e-mail depuis la BDD.

\*/

function getGabaritEmail(idGabarit, langueCode) {

const systemIds = getSystemIds();

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

const gabaritsSheet = bdd.getSheetByName("Gabarits\_Emails");

if (!gabaritsSheet) throw new Error("L'onglet 'Gabarits\_Emails' est introuvable.");

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

const headers = data.shift();

const idCol = headers.indexOf('ID\_Gabarit');

const langCol = headers.indexOf('Langue');

const gabaritRow = data.find(row => row[idCol] === idGabarit && row[langCol].toUpperCase() === langueCode.toUpperCase());

if (!gabaritRow) throw new Error(`Aucun gabarit trouvé pour l'ID '${idGabarit}' et la langue '${langueCode}'.`);

const gabarit = {};

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

if (header) { gabarit[header] = gabaritRow[index]; }

});

return gabarit;

}

/\*\*

\* CORRIGÉ : Formate le bloc de scores détaillés en fonction du niveau requis.

\*/

function formatScoresDetails(resultats, niveauDetails, typeTest, langueCode) {

if (niveauDetails === 'Simple') {

return "";

}

let scoresText = "";

const T = loadTraductions(langueCode);

if (niveauDetails === 'Détaillé' && resultats.scoresData) {

if (typeTest === 'Couleurs' || typeTest === 'Ancres') {

scoresText += (T[`INTRO\_SCORES\_DETAILLES\_${typeTest.toUpperCase()}`] || "Voici le détail de vos scores :") + "\n";

Object.entries(resultats.scoresData).sort((a, b) => b[1] - a[1]).forEach(([code, score]) => {

scoresText += `- ${resultats.mapCodeToName[code] || code} : ${score} ${T.SUFFIXE\_POINTS || 'points'}\n`;

});

} else if (typeTest === 'MBTI') {

scoresText += (T.INTRO\_SCORES\_DETAILLES\_MBTI || "Voici le détail de vos préférences :\n");

const s = resultats.scoresData;

scoresText += `- ${(T.AXE\_EI || "Extraversion (E) vs Introversion (I)")} : ${s.E} vs ${s.I}\n`;

scoresText += `- ${(T.AXE\_SN || "Sensation (S) vs Intuition (N)")} : ${s.S} vs ${s.N}\n`;

scoresText += `- ${(T.AXE\_TF || "Pensée (T) vs Sentiment (F)")} : ${s.T} vs ${s.F}\n`;

scoresText += `- ${(T.AXE\_JP || "Jugement (J) vs Perception (P)")} : ${s.J} vs ${s.P}\n`;

}

}

return scoresText + "\n";

}

/\*\*

\* MODIFIÉ : Construit et envoie les e-mails en utilisant les gabarits de la BDD.

\*/

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

try {

const idGabarit = config.ID\_Gabarit\_Email\_Repondant;

if (!idGabarit) {

throw new Error("La colonne 'ID\_Gabarit\_Email\_Repondant' n'est pas définie dans la configuration du test.");

}

const gabarit = getGabaritEmail(idGabarit, langueCode);

const T = loadTraductions(langueCode);

const variables = {

nom\_repondant: reponse.nomRepondant || 'Participant',

Type\_Test: config.Type\_Test || '',

profilFinal: resultats.titreProfil || resultats.profilFinal || '',

formateur\_nom: config.Formateur\_Nom || 'Votre Formateur',

scoresData: formatScoresDetails(resultats, gabarit.Niveau\_Details\_Resultats, config.Type\_Test, langueCode)

};

let corpsEmail = "";

corpsEmail += (gabarit.Corps\_Salutation || "") + "\n";

corpsEmail += (gabarit.Corps\_Introduction || "") + "\n\n";

corpsEmail += (gabarit.Corps\_Entete\_Resultats || "") + "\n\n";

corpsEmail += (gabarit.Corps\_Conclusion || "") + "\n\n";

corpsEmail += (gabarit.Corps\_Signature || "");

let sujet = gabarit.Sujet;

for (const [key, value] of Object.entries(variables)) {

const regex = new RegExp(`\\{${key}\\}`, 'g');

sujet = sujet.replace(regex, value);

corpsEmail = corpsEmail.replace(regex, value);

}

corpsEmail = corpsEmail.replace('\_{scoresData}\_', variables.scoresData);

const piecesJointes = findAttachments(config.Type\_Test, resultats.profilFinal, gabarit.Niveau\_Pieces\_Jointes, langueCode);

const adressesUniques = new Set();

if (config.Repondant\_Email\_Actif === 'Oui' && reponse.emailRepondant) { adressesUniques.add(reponse.emailRepondant); }

if (config.Patron\_Email\_Mode === 'Oui' && config.Patron\_Email) { adressesUniques.add(config.Patron\_Email); }

if (config.Formateur\_Email\_Actif === 'Oui' && config.Formateur\_Email) { adressesUniques.add(config.Formateur\_Email); }

if (config.Developpeur\_Email) { adressesUniques.add(config.Developpeur\_Email); }

if (isDebugMode) { LoggerLib.logEmailQueue(adressesUniques); }

adressesUniques.forEach(adresse => {

try {

let sujetFinal = sujet;

if (adresse.toLowerCase() !== (reponse.emailRepondant || "").toLowerCase()) {

sujetFinal = (T.PREFIXE\_COPIE\_EMAIL || "Copie : ") + sujet;

}

if (isDebugMode) { LoggerLib.logEmailAttempt(adresse, sujetFinal); }

MailApp.sendEmail({

to: adresse,

subject: sujetFinal,

body: corpsEmail,

attachments: piecesJointes

});

if (isDebugMode) { LoggerLib.logSuccess(`Envoi réussi à ${adresse}`); }

} catch (e) {

if (isDebugMode) { LoggerLib.logError(`Échec de l'envoi à ${adresse}`, e); }

}

});

if (isDebugMode) { LoggerLib.logEnd(); }

} catch (err) {

if (typeof LoggerLib !== 'undefined' && isDebugMode) { LoggerLib.logError("Erreur critique dans buildAndSendEmails", err); }

Logger.log("ERREUR CRITIQUE V4 : " + err.toString() + "\n" + err.stack);

}

}

/\*\*

\* CORRIGÉ (v4.2) : Trouve les PJ en utilisant l'en-tête 'Email\_Niveau'.

\*/

function findAttachments(typeTest, profilCode, niveauPJ, langueCode) {

try {

const systemIds = getSystemIds();

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

const pjSheet = bdd.getSheetByName("sys\_PiecesJointes");

if (!pjSheet) { return []; }

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

const headers = data.shift();

const idx = {

type: headers.indexOf('Type\_Test'),

profil: headers.indexOf('Profil\_Code'),

niveau: headers.indexOf('Email\_Niveau'),

langue: headers.indexOf('Langue'),

id: headers.indexOf('ID\_Fichier\_Drive')

};

if (Object.values(idx).some(i => i === -1)) {

Logger.log("Avertissement : une ou plusieurs colonnes sont manquantes dans 'sys\_PiecesJointes'. Colonnes attendues : Type\_Test, Profil\_Code, Email\_Niveau, Langue, ID\_Fichier\_Drive.");

return [];

}

const niveauNumRequis = parseInt(niveauPJ.replace(/[^0-9]/g, ''), 10) || 1;

const idsFichiersTrouves = new Set();

data.forEach(row => {

const typeMatch = (row[idx.type] === typeTest);

const profilMatch = (row[idx.profil] === profilCode || (typeof row[idx.profil] === 'string' && row[idx.profil].toUpperCase() === 'TOUS'));

const niveauMatch = (row[idx.niveau] > 0 && row[idx.niveau] <= niveauNumRequis);

const langueMatch = (row[idx.langue].toUpperCase() === langueCode.toUpperCase() || row[idx.langue].toUpperCase() === 'TOUS');

if (typeMatch && profilMatch && niveauMatch && langueMatch && row[idx.id]) {

idsFichiersTrouves.add(row[idx.id]);

}

});

if (idsFichiersTrouves.size === 0) return [];

const fichiers = [];

idsFichiersTrouves.forEach(id => {

try {

fichiers.push(DriveApp.getFileById(id).getBlob());

} catch (e) {

Logger.log(`Impossible d'accéder au fichier Drive avec l'ID : ${id}`);

}

});

return fichiers;

} catch (e) {

Logger.log(`Erreur critique dans findAttachments : ${e.toString()}`);

return [];

}

}

// Fonctions inchangées

function loadTraductions(langueCode) {

if (!langueCode) {

throw new Error("Le code de langue fourni à loadTraductions est indéfini.");

}

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 && h.toLowerCase() === langueCode.toLowerCase());

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

const traductions = {};

const keyColIndex = 0;

data.forEach(row => {

if (row[keyColIndex]) { traductions[row[keyColIndex]] = row[langColIndex]; }

});

return traductions;

}

function mapQuestionsById(bdd, nomFeuille) {

const sheet = bdd.getSheetByName(nomFeuille);

if (!sheet) { throw new Error(`Feuille de questions '${nomFeuille}' introuvable.`); }

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

const headers = data.shift();

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

const paramsCol = headers.indexOf('Paramètres (JSON)');

const mapById = {};

data.forEach(row => {

const qId = row[idCol];

if (qId) {

mapById[qId] = { id: qId, params: row[paramsCol] };

}

});

return mapById;

}

function getOriginalLanguage(reponses) {

const langueRepondantBrute = reponses['Langue / Language'] ? reponses['Langue / Language'][0] : 'Français';

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

return mapLangue[langueRepondantBrute] || 'FR';

}

**5.2. Architecture Détaillée et Traçabilité**

* **[LIBRAIRIE] LoggerLib (Apps Script) (INCHANGÉ)**
* **[TEMPLATE]V2 Kit de Traitement (Projet Apps Script)**
  + TraitementReponses.gs (INCHANGÉ)
  + **Utilities.gs (MODIFIÉ)**
    - **buildAndSendEmails (MODIFIÉ)** : Logique entièrement réécrite pour utiliser les gabarits. La fonction est maintenant un orchestrateur qui assemble l'e-mail à partir des blocs de texte de la BDD.
    - **findAttachments (MODIFIÉ)** : La logique a été corrigée pour utiliser l'en-tête Email\_Niveau et pour inclure les pièces jointes des niveaux inférieurs ou égaux au niveau demandé.
    - **getGabaritEmail (AJOUTÉ)** : Nouvelle fonction utilitaire pour récupérer une ligne de gabarit depuis l'onglet Gabarits\_Emails.
    - **formatScoresDetails (AJOUTÉ)** : Nouvelle fonction utilitaire pour formater le bloc de texte des scores détaillés, isolant la logique spécifique à chaque test.
  + Logique\_Universel.gs (INCHANGÉ)

**5.3. Nomenclature des Variables**

| Nom Précis | Type de Donnée | Origine/Portée | Contenu Attendu & Structure | Exemple Concret |
| --- | --- | --- | --- | --- |
| idGabarit | String | buildAndSendEmails (Utilities.gs) | L'ID du gabarit à utiliser, lu depuis [CONFIG]. | "RESULTATS\_N2" |
| gabarit | Object | buildAndSendEmails (Utilities.gs) | Objet JS contenant toutes les données de la ligne du gabarit. | { ID\_Gabarit: "RESULTATS\_N2", Sujet: "Analyse...", ... } |
| variables | Object | buildAndSendEmails (Utilities.gs) | Objet clé-valeur de tous les placeholders à injecter dans le texte. | { nom\_repondant: "Victor", Type\_Test: "Couleurs", ... } |
| niveauPJ | String | findAttachments (Utilities.gs) | Le code du niveau de pièces jointes requis. | "N1", "N2" |

Exporter vers Sheets

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

* **Fichier : [CONFIG]V2 Usine à Tests**
  + **Onglet Paramètres Généraux** : ... (colonnes existantes), **ID\_Gabarit\_Email\_Repondant (AJOUTÉ)**, **Formateur\_Nom (AJOUTÉ)**. La colonne Repondant\_Contenu est désormais obsolète.
* **Fichier : [BDD] Tests & Profils**
  + **Onglet Gabarits\_Emails (AJOUTÉ)** : ID\_Gabarit, Langue, Sujet, Corps\_Salutation, Corps\_Introduction, Corps\_Entete\_Resultats, Corps\_Conclusion, Corps\_Signature, Niveau\_Details\_Resultats, Niveau\_Pieces\_Jointes.
  + **Onglet sys\_PiecesJointes** : Type\_Test, Profil\_Code, Email\_Niveau, Langue, Nom\_Pour\_Info, ID\_Fichier\_Drive.

**6. Actions Recommandées / Prochaines Étapes**

* **Action Immédiate :** Démarrer le chantier N°2 en s'attaquant au fichier TraitementReponses.gs.
* **Prochaine Session :**
  1. Modifier la fonction retraiterReponse\_UI dans TraitementReponses.gs pour qu'elle ouvre une interface HTML au lieu de lancer directement le traitement.
  2. Discuter de la conception de cette interface (popup vs. barre latérale) et des informations qu'elle doit afficher et permettre de modifier.
  3. Créer une première version du fichier RetraitementUI.html.