

PHOTO LAB MASTER — PLM-MJML

PLM-MJML_2026-10-08_DOCUMENT-MAITRE-ARCHITECTURE-CONVERSION-SPIP-V004.docx

DOCUMENT MAÎTRE CANONIQUE | Architecture, imports QuarkXPress et InDesign, HTML → MJML, édition visuelle, intégration SPIP et feuille de route

Fichier canonique — PLM-MJML_2026-10-08_DOCUMENT-MAITRE-ARCHITECTURE-CONVERSION-SPIP-V004.docx

Version / date — V004 — 8 octobre 2026 | Remplace V002 comme proposition de référence, sans écrasement de V001/V002

Statut — PROPOSITION TECHNIQUE — NON INSTALLÉE — VALIDATION REQUISE AVANT TOUTE MODIFICATION PLM

Destination pressentie — /Users/adam/A_PHOTO-LAB-MASTER/PLM-MJML/ (à confirmer après vérification de la BIBLE et des README ; ne pas créer automatiquement)

Périmètre : architecture complète multi-entrées, éditeur GrapesJS, double moteur HTML/MJML, connecteurs InDesign/QuarkXPress et intégration SPIP ; spécification sans déploiement.

0. Référentiel, gouvernance et prérequis

Le présent document est une spécification autonome. La BIBLE PHOTO LAB MASTER active, ses empreintes, et les README effectifs du dossier maître, de la destination et des dépendances ne sont pas accessibles ici : aucun statut « conforme au canon local » n'est donc prétendu. Avant tout dépôt local : vérifier l'état réel, lire BIBLE et README, auditer en lecture seule, établir un dry-run, obtenir une validation explicite, enregistrer état antérieur et plan de retour, appliquer puis contrôler.

Garde-fous — Aucun écrasement silencieux ; aucun travail dans Downloads comme emplacement PLM final ; pas de choix de la version canonique sur le seul nom ; aucune modification de base de données, dossier ou serveur sans validation.

1. Objectif fonctionnel

Produire un système SPIP-MJML capable d'importer une newsletter créée dans QuarkXPress, Adobe InDesign ou en HTML ; de la convertir vers un modèle de courriel responsive, structurellement éditable ; de proposer une révision visuelle de type What You See Is What You Get (WYSIWYG, « ce que vous voyez est ce que vous obtenez ») ; puis de publier ou envoyer via l'écosystème SPIP.

Le résultat doit préserver séparément (a) les sources, (b) un modèle intermédiaire normalisé en JavaScript Object Notation (JSON), (c) le code Mailjet Markup Language (MJML), (d) le HyperText Markup Language (HTML) compilé, (e) les vues de référence, et (f) une trace de conversion et validation.

2. Principes et limitations structurelles

- EPS (Encapsulated PostScript) et PDF (Portable Document Format) ne sont PAS des formats pivots : ils fournissent des références visuelles, non une structure éditoriale fiable.
- La fidélité pixel à pixel sur tous les clients de messagerie est impossible à garantir : Outlook, Apple Mail et Gmail n'utilisent pas un moteur de rendu identique.
- Un document imprimé à colonnes et positions absolues devra être reconfiguré dans une grille de courriel à colonnes réductibles et ordre de lecture explicite.
- Le mode « fidélité visuelle » peut utiliser des visuels rasterisés, avec perte d'accessibilité et d'édition. Le mode « responsive éditable » est prioritaire.

- Aucune conversion HTML arbitraire → MJML ne peut être sans perte : l'import doit produire une liste explicite de propriétés abandonnées ou approximées.
- Le JSON intermédiaire est un schéma PLM propre et versionné, distinct du format JSON éventuellement accepté par des outils MJML.

3. Entrées et chemins de conversion

Entrée	Export recommandé	Priorité	Remarques
Adobe InDesign	IDML (InDesign Markup Language) + HTML / HTML5	P1 / P2	IDML riche mais parser complexe ; HTML pour pilote rapide
QuarkXPress	HTML5 Publication ou EPUB (Electronic Publication)	P1 / P2	Évaluer poids des styles et éléments interactifs
HTML externe	HTML + feuilles CSS + images	P0	Point de départ et socle universel
EPS / PDF	PNG de contrôle + inspection	Référence	Ne sert pas d'entrée principale
MJML existant	MJML source	P0	Validation, compilation et édition

4. Pipeline de données canonique

SOURCE ORIGINALE (lecture seule) → extraction sécurisée → manifeste assets → DOM/CSS/IDML normalisé → JSON PLM-EMAIL-DOC V001 → mappage sémantique en composants MJML → validation MJML → compilation HTML → insertion/interprétation SPIP selon stratégie testée → contrôles d'affichage → revue humaine → modèle publiable.

DOM = Document Object Model (modèle objet du document) ; CSS = Cascading Style Sheets (feuilles de style en cascade). Une transformation XHTML/HTML vers MJML est une reconstruction sémantique, non un renommage de balises.

5. Schéma du document intermédiaire

Entités minimales : document_id, source_type, source_sha256, schema_version, title, language, canvas_width, metadata, sections[], blocks[], assets[], warnings[], provenance[], approvals[]. Chaque bloc contient : block_id, type, text, source_reference, style_normalized, responsive_policy, accessibility, confidence, conversion_notes.

Types de blocs : heading, paragraph, image, columns, spacer, divider, table, button, hero, raw_fallback et dynamic_spip. Les images comprennent les dimensions, nom de fichier, empreinte SHA-256 (Secure Hash Algorithm, sortie de 256 bits), texte de remplacement, licence, URI source et URI publique.

Contrat : identifiants stables ; versionnage explicite ; ordonnancement de lecture ; rejet des ressources distantes non autorisées ; journal des pertes de mise en page ; références persistantes à la source. La structure reste réversible vers MJML ; le retour exact vers InDesign ou QuarkXPress n'est pas promis.

6. Convertisseur HTML → MJML : algorithme détaillé

1. Recevoir un paquet HTML ou HTML5 contenant code, CSS, fichiers d'images, manifeste et origine ; stocker un exemplaire immuable.
2. Valider MIME, taille, noms et arborescence ; extraire les archives dans un espace isolé ; bloquer liens symboliques dangereux, parcours de chemin et fichiers exécutables.
3. Parser le HTML sans exécuter JavaScript ; supprimer scripts, formulaires, événements inline et intégrations actives.
4. Résoudre les feuilles de style CSS autorisées et calculer les styles utiles : police, couleur, largeur, marges, alignement, arrière-plan et hiérarchie.

5. Reconstruire le flux éditorial : headings, paragraphes, images, liens, listes, tableaux simples et regroupements en sections.
6. Dédire les colonnes à partir de la hiérarchie et des mesures, avec règles explicites de résolution des ambiguïtés et ordre mobile.
7. Établir un JSON versionné et un rapport d'éléments non transposables (position absolue, animations, scripts, flottants, formulaires, vidéos).
8. Mapper JSON → composants MJML (mj-section, mj-column, mj-text, mj-image, mj-button, mj-divider, mj-spacer, mj-table) et choisir un fallback documenté.
9. Valider la syntaxe MJML, compiler en HTML et contrôler les erreurs et avertissements.
10. Comparer visuellement le rendu à la source, vérifier texte, liens, ordre mobile et images ; présenter les écarts à correction humaine.
11. Enregistrer sources, sorties, paramètres, empreintes, version du compilateur, moteur, date et décisions de validation.

7. Import InDesign et QuarkXPress : modes opératoires

7.1 Adobe InDesign — pilote via HTML

12. Préparer styles de paragraphe/objet explicites et séquence de lecture ; vérifier que les images restent liées et exploitables.
13. Exporter en HTML ou HTML5 ; conserver le paquet CSS et les images, ainsi qu'un PDF de référence.
14. Importer HTML dans PLM ; convertir vers JSON puis MJML ; signaler les objets non importés.
15. Réviser manuellement colonnes, polices, couleurs, ordre mobile et liens ; figer le modèle approuvé.

7.2 Adobe InDesign — import avancé IDML

16. Exporter une copie IDML et inventorier les fichiers contenus dans le paquet.
17. Analyser les stories, spreads, ressources, styles et liens, avec interpréteur propre à la version du format.
18. Reconstituer l'ordre de lecture indépendamment des coordonnées physiques ; rapprocher objets et contenu textuel.
19. Faire correspondre styles source et styles sémantiques du JSON, avec table de règles réutilisable.
20. Comparer aux exports HTML et PDF ; conserver divergences et produire le MJML après validation.

7.3 QuarkXPress — import HTML5 / EPUB

21. Exporter une HTML5 Publication ou EPUB avec les images accessibles et une référence PDF.
22. Identifier la page ou l'article souhaité : ne pas importer aveuglément l'ensemble des éléments interactifs du paquet.
23. Récupérer XHTML/HTML, styles, médias et ordre des contenus.
24. Normaliser vers JSON ; mapper en MJML ; reconstruire manuellement toute disposition non exprimable en courriel.
25. Valider visuellement sur formats bureau et mobile ; enregistrer le profil QuarkXPress employé.

8. Plugin SPIP-MJML : modules

Module	Rôle	Priorité
PLM-MJML-CORE	Schéma JSON, audit, validation et compilation	P0
PLM-MJML-HTML-IMPORT	Import HTML, CSS et médias	P0
PLM-MJML-EDITOR	Code source + éditeur visuel de blocs	P1
PLM-MJML-SPIP	Interface espace privé et prévisualisation SPIP	P1
PLM-MJML-INDESIGN	Connecteur IDML avancé	P2
PLM-MJML-QUARK	Connecteur HTML5 / EPUB	P2

PLM-MJML-QA	Régression visuelle, clients e-mail et journaux	P1
PLM-MJML-RELEASE	Signatures, migrations, versionnage, rollback	P1

Structure SPIP indicative : paquet.xml ; spip_mjml_administrations.php ; formulaires/editer_mjml.html et .php ; prive/squelettes/contenu/mjml.html ; actions/compiler_mjml.php ; inc/mjml_compiler.php ; lang/spip_mjml_fr.php, _de.php, _it.php, _en.php ; assets ; tests. paquet.xml définit versions, permissions et dépendances après vérification de la version SPIP en production.

9. Pipeline MJML / SPIP : précautions critiques

Stratégie par défaut : conserver les données dynamiques en emplacements identifiés, compiler une enveloppe MJML sûre en HTML, puis interpréter le squelette final côté SPIP. Pour des boucles qui entourent plusieurs composants MJML, employer mj-raw uniquement si la conservation syntaxique et la compilation ont été démontrées en tests ; sinon utiliser des jetons temporaires opaques et une restauration strictement contrôlée.

Le moteur ne doit jamais interpréter des boucles arbitraires soumises par un utilisateur non autorisé. Les droits de rédaction et de modification des modèles sont séparés ; protection contre la falsification des requêtes intersites ; assainissement de la prévisualisation et isolation des imports.

Newsletters : modèles à intégrer selon l'interface effective du plugin installé ; Facteur pour la livraison ; Mailshot si le mode d'envoi le requiert. L'existence d'un plugin n'autorise pas à supposer les chemins ou signatures de fonction : tester sur site de développement.

10. Compilation et hébergement

Option A — compilation locale Node.js en processus isolé, hors requêtes publiques ; recommandée si les capacités exactes du serveur le permettent. Option B — service de compilation distinct contrôlé, communiquant avec SPIP en PHP (PHP: Hypertext Preprocessor) via API (Application Programming Interface). Option C — API externe MJML uniquement sur décision explicite liée à la confidentialité, au coût et à la disponibilité.

Interdire appels réseau des fichiers importés ; limiter mémoire et temps de traitement ; versionner le moteur et les dépendances ; enregistrer toutes les commandes reproductibles. L'hébergement Infomaniak et ses droits d'exécution doivent être audités avant de choisir le mode réel.

11. Éditeur visuel WYSIWYG : spécification

- Panneau de gauche : bibliothèque de blocs (titre, texte, image, colonnes, bouton, espacement, séparateur).
- Panneau central : rendu de travail bureau et mobile ; déplacement, redimensionnement limité aux contraintes MJML ; aucune promesse de positionnement libre de type PAO.
- Panneau de droite : propriétés d'un bloc (police, largeur, fond, marges, lien, texte alternatif, comportement mobile).
- Mode double vue : source originale / résultat compilé ; superposition semi-transparente et rapport de différences.
- Mode expert : édition du MJML et des boucles SPIP avec validation immédiate, archivage des versions et possibilité de retour.
- Internationalisation française, allemande, italienne et anglaise ; prise en charge des variantes de contenu sans dupliquer aveuglément tous les styles.

12. Feuille de route versionnée et critères de sortie

Version	Livrable / opérations	Condition pour franchir
V000 — audit	Inventaire SPIP, serveur, versions, licences, BIBLE/README, jeu d'essai	Aucun changement ; rapport validé
V001 — socle	Schéma JSON, import HTML, tests d'analyse et export MJML	10 documents tests ; aucun texte perdu sans alerte

V002 — compilation	MJML → HTML, sauvegarde des sources, rapports, prévisualisation	Rendus reproductibles ; erreurs bloquantes signalées
V003 — SPIP	Plugin installable, permissions, formulaires, squelettes et test Newsletters	Tests sur SPIP de développement ; aucun envoi réel involontaire
V004 — InDesign HTML	Profil InDesign HTML/HTML5, styles et assets	Cas documentés, ordre de lecture vérifié
V005 — QuarkXPress	Profil HTML5/EPUB ; gestion média et extraction	Cas documentés, avertissements complets
V006 — IDML	Extraction structurée IDML et règles sémantiques	Concordance source/JSON audité
V007 — WYSIWYG	Éditeur de blocs et comparaison visuelle bureau/mobile	Relecture humaine et historique opérationnels
V008 — production pilote	Sécurité, sauvegardes, rollback, mail clients, validation	Publication contrôlée après autorisation
V009 — industrialisation	Import par lots, profils, traduction, métriques, migration	Mesures de robustesse et dossiers de preuves

13. Jeu d'essai et mesure de qualité

Créer au moins 10 maquettes de complexité croissante : 1 colonne, 2 colonnes, galerie de photographies, articles SPIP, mise en page avec légendes, tableaux, langues multiples, colonnes asymétriques, styles complexes et document très long. Pour chaque cas : sources/export, PDF référence, JSON attendu, MJML, HTML, images de rendu, journal des différences, validation humaine.

Mesures : conservation du texte et liens (objectif 100 % ou erreur explicitement signalée), taux d'assets associés, proportion de blocs éditables, nombre de corrections humaines, écarts de rendu desktop/mobile, erreurs MJML, temps de conversion, compatibilité des principaux clients mail. Ne pas fixer un score visuel universel sans corpus de référence et protocole de mesure.

14. Sécurité, licences et exploitation

- Conserver le fichier original immuable avec empreinte SHA-256 et registre des transformations.
- Distinguer licences du code du plugin, du moteur MJML, des bibliothèques utilisées, des polices, et des images ; conserver les preuves de droits.
- Éviter toute exécution d'HTML, de JavaScript, de macro ou de fichier ajouté par l'importeur.
- Gestion des droits : administrateur, concepteur de modèles, rédacteur, réviseur, responsable de publication.
- Journaliser importation, modification, compilation, publication, et référence de retour à la version précédente.
- Ne pas expédier d'images intégrées volumineuses : utiliser URLs HTTPS (Hypertext Transfer Protocol Secure) permanentes et textes de remplacement ; tester l'affichage avec images bloquées.
- Gérer désinscription, mentions légales et consentement selon les obligations applicables ; ne jamais fusionner automatiquement les contacts depuis les outils de PAO.

15. Mode opérateur de réalisation et livraison

- Vérifier la BIBLE et les README canoniques actifs ; relever version, empreinte et état des dossiers PLM concernés.
- Auditer en lecture seule l'installation SPIP, Newsletters, Facteur, Mailshot, version PHP et modalités éventuelles d'exécution de Node.js.
- Réunir les maquettes et exports tests sous un dossier de travail distinct des sources immuables.
- Rédiger un dry-run nommant exactement dossiers à créer, fichiers, tables éventuelles, tâches, permissions et procédure de retour.
- Obtenir la validation explicite du dry-run ; aucune installation ou mutation préalable.
- Développer le convertisseur HTML isolément et obtenir un corpus de résultats attestés.

32. Développer paquet.xml et l'interface SPIP dans un environnement de staging ; tester compilation et interprétation de boucles.
33. Déployer les profils InDesign puis QuarkXPress ; ajouter ultérieurement IDML et éditeur WYSIWYG complet.
34. Générer manifeste, hash, rapport de tests, documentation de l'administrateur et du rédacteur, et procédure de rollback.
35. Effectuer une revue de préproduction avant déploiement et un post-contrôle après approbation.

16. Arborescence PROPOSÉE (pas encore créée)

PLM-MJML/

README.md

00-GOVERNANCE/ (manifestes, décisions, versions, rollback)

01-SOURCES/ (copies contrôlées, lecture seule)

02-TEST-FIXTURES/ (HTML, IDML, Quark, rendus de référence)

03-SCHEMA-JSON/

04-CONVERTER-HTML/

05-CONVERTER-INDESIGN/

06-CONVERTER-QUARKXPRESS/

07-SPIP-PLUGIN/

08-WYSIWYG-EDITOR/

09-QA-REPORTS/

10-RELEASES/

Cette arborescence doit être comparée à la nomenclature des dossiers de production et des dossiers sémantiques déjà fixée par la gouvernance PLM ; elle ne vaut pas autorisation de création.

17. Registre des décisions en suspens

D01 — Version SPIP réelle et versions des plugins Newsletter/Facteur/Mailshot

D02 — Dossier propriétaire canonique validé par BIBLE/README

D03 — Compilateur local sur serveur, service interne ou API externe

D04 — Formats exacts QuarkXPress et InDesign présents dans le corpus

D05 — Choix de l'éditeur visuel et modalités de licence

D06 — Mode de stockage (fichiers, tables SPIP, conservation source et versions)

D07 — Périmètre de publication multilingue et autorisations

D08 — Critères d'acceptation et conformité du rendu sur les clients e-mail

18. Sources primaires consultées (8 octobre 2026)

MJML — référence composants et mj-raw — <https://documentation.mjml.io/>

MJML — interface en ligne de commande — <https://github.com/mjml/mjml/blob/master/packages/mjml-cli/README.md>

Adobe — InDesign HTML5 — <https://helpx.adobe.com/fr/indesign/desktop/save-export-and-publish/export-to-html-and-web/export-content-as-html5.html>

Adobe — export HTML hérité — <https://helpx.adobe.com/fr/indesign/desktop/save-export-and-publish/export-to-html-and-web/export-to-html-legacy.html>

Adobe — fichiers IDML — <https://helpx.adobe.com/indesign/desktop/save-export-and-publish/save-and-export/save-documents.html>

Adobe — export EPUB — <https://helpx.adobe.com/indesign/desktop/save-export-and-publish/export-to-epub/export-to-epub.html>

Quark — HTML5 Publication —

https://www.quark.com/documentation/quarkxpress/2026/english/Digital%20Publishing%20with%20QuarkXPress%202026/_7bb86344-83c1-4d6b-8e9b-8e11743d70ca.html

Quark — options EPUB et HTML5 —

https://www.quark.com/documentation/quarkxpress/2026/english/Digital%20Publishing%20with%20QuarkXPress%202026/_f431b954-19a7-418c-b7a8-9c9d41c22bad.html

SPiP — paquet.xml — <https://plugins.spip.net/redaction-du-paquet-xml.html>

SPiP — pipelines — <https://programmer.spip.net/Utiliser-les-pipelines>

Fin de la spécification initiale V001, conservée dans V002. Prochaine étape conditionnelle : audit canonique en lecture seule, puis dry-run documenté ; aucun APPLY autorisé par la seule existence de ce document.

19. Addendum V002 — périmètre et statut

La version V002 reprend sans suppression les sections 0 à 18 de V001 et ajoute le chiffrage, les lots de développement, les opérations de migration et la conception de la deuxième génération fonctionnelle. Les estimations sont indicatives, non contractuelles. Le présent document ne signifie ni installation, ni modification du système de fichiers PHOTO LAB MASTER.

Nomenclature : « V001/V002 » désignent les versions de ce document maître ; « produit 0.1 à 0.9 » désigne les versions de livraison du plugin. La « version 2 » fonctionnelle (produit 2.0) désigne la seconde génération après stabilisation du produit initial.

20. Estimation d'effort, dépendances et budget-temps

Lot	Effort (jours ouvrés)	Livrable mesurable
Audit canonique, SPiP, dépendances	1-2	Inventaire + décisions techniques
Socle plugin SPiP (administration, stockage, droits)	2-3	Plugin installable en préproduction
Compilation MJML vers HTML	1-2	Compiler isolé + journaux
Importation et reconstruction HTML → MJML	3-5	Parseur + rapport de pertes
Passerelle Adobe InDesign	3-5	Import HTML/IDML avec fixtures
Passerelle QuarkXPress	3-5	Import HTML5/EPUB avec fixtures
Éditeur visuel et comparateur responsive	3-5	Édition des blocs structurés
Multilingue + Newsletters/Facteur/Mailshot	1-2	Essais SPiP FR/DE/IT/EN
Tests transversaux, sécurité, documentation	3-6	Campagne de validation reproductible

Total de planification : 20 à 35 jours ouvrés avec un développeur expérimenté et assistance par intelligence artificielle, soit environ 4 à 7 semaines à temps plein. Prévoir 5 à 10 jours de réserve pour les cas difficiles (typographies, positions absolues, Outlook, formats d'export et droits de diffusion). Cible de gestion prudente : 30 jours + réserve, soit environ 7 à 8 semaines calendaires. La somme des extrema des lignes du tableau n'est pas un engagement : certains travaux se recouvrent, et les arbitrages changent la charge.

Niveau	Horizon indicatif	Périmètre
--------	-------------------	-----------

Prototype technique	5–8 jours	Import HTML simple ; JSON intermédiaire ; MJML ; aperçu
Version utilisable	12–20 jours	Plugin SPIP ; flux éditorial ; passerelle InDesign initiale
V0.9 stabilisée	20–35 jours	Passerelle QuarkXPress ; édition ; tests ; multilingue
Produit 2.0	+15–30 jours	Fonctions avancées décrites au chapitre 23 ; après validation terrain

21. Plan de réalisation V000 → V009

Version produit	Opérations / critères de sortie
V000 — Audit	Lire BIBLE active, README, paquetages, versions SPIP ; relever hébergement, outils, licences ; dry-run sans écriture
V001 — Noyau	Paquet SPIP, routage privé, permissions, configuration, modèle de données, audit
V002 — Compilation	Service MJML → HTML, diagnostic de syntaxe, compilation déterministe, source immuable
V003 — HTML	Parseur HTML/CSS, extraction assets, mapping MJML, niveau de confiance par bloc
V004 — InDesign	Export IDML/HTML, typographie, images liées, ordre de lecture, jeu de tests
V005 — QuarkXPress	Export HTML5/EPUB, adaptation des blocs, légendes, images et texte
V006 — WYSIWYG	Édition structurée, rendu bureau/mobile, aller-retour JSON sans perte interne
V007 — SPIP	Boucles protégées, langues, gestion des articles, test Newsletters/Facteur/Mailshot
V008 — Qualité	Tests multi-clients, accessibilité, sécurité, performance, non-régression
V009 — Livraison	Documentation opérateur, support, journal d'installation, rollback, validation de publication

22. Procédures détaillées de réalisation

22.1 Import et inventaire

- 1 — Copier le document source dans une zone de traitement isolée, conserver l'original en lecture seule et calculer SHA-256 (Secure Hash Algorithm, empreinte cryptographique sur 256 bits).
- 2 — Exporter depuis InDesign en HTML ou IDML (InDesign Markup Language) ; depuis QuarkXPress en HTML5 ou EPUB (Electronic Publication). Exporter une référence PDF (Portable Document Format) pour comparaison visuelle.
- 3 — Lister polices, images, liens, métadonnées, dimensions, texte masqué, couleurs et éléments dépassant du cadre. Signaler les polices non distribuables et les images absentes.
- 4 — Extraire HTML, CSS (Cascading Style Sheets), contenus textuels et fichiers associés ; interdire les scripts actifs et URLs non autorisées ; neutraliser les entrées non fiables.

22.2 Interprétation de la maquette et conversion

- 5 — Construire un arbre intermédiaire PLM-EMAIL-DOC versionné, avec identifiants stables, sémantique des blocs, sources et coordonnées de référence.

- 6 — Reconnaître titres, paragraphes, images, colonnes, tableaux, boutons, légendes et séparateurs ; définir explicitement l'ordre de lecture mobile.
- 7 — Mapper les éléments aux composants mj-section, mj-column, mj-text, mj-image, mj-button et mj-table lorsque possible ; journaliser chaque propriété approximée ou abandonnée.
- 8 — Refuser toute reproduction « pixel exacte » comme garantie ; distinguer le mode graphique (blocs image) du mode éditale responsive, en recommandant ce dernier.

22.3 SPIP, compilation et publication

- 9 — Stocker source, JSON, MJML, assets et versions avec provenance ; n'appliquer la compilation qu'à une version approuvée.
- 10 — Protéger les boucles et balises SPIP via jetons contrôlés ou mj-raw suivant leur emplacement ; vérifier leur restauration exacte après compilation.
- 11 — Compiler MJML en HTML ; produire un rapport erreurs/avertissements ; faire ensuite interpréter le squelette par SPIP dans un environnement contrôlé.
- 12 — Remplacer tous les liens relatifs d'images et ressources par des liens absolus autorisés ; ajouter texte alternatif, lien de désinscription et mentions requis pour la liste concernée.
- 13 — Générer prévisualisations bureau et mobile ; réaliser des essais d'envoi sur boîtes de test ; n'activer Mailshot que pour un lot autorisé.

22.4 Assurance qualité et retour arrière

- 14 — Comparer référence PDF et HTML rendu : captures, superpositions, hiérarchie, lisibilité, liens, images manquantes ; consigner les écarts acceptés.
- 15 — Tester Gmail, Apple Mail et Outlook (versions représentatives) ; vérifier images bloquées, dark mode, affichage mobile et texte alternatif.
- 16 — Valider des fixtures unitaires : HTML simple/complex, IDML, EPUB, langues FR/DE/IT/EN, caractères spéciaux, contenus invalides, fichiers volumineux.
- 17 — Consigner version, hash, responsable, approbation, date et paquet à restaurer ; sauvegarder la version précédente et tester effectivement le rollback.

23. Version fonctionnelle 2.0 — fonctionnalités envisagées

Le produit 2.0 n'est pas inclus dans l'estimation initiale de 20-35 jours. Les fonctions suivantes sont des candidats soumis à arbitrage après résultats mesurés sur des maquettes réelles.

Priorité	Module V2	Spécification et validation
P1	Import en lots et file de travaux	Dossiers de sources, progression, reprise après erreur, rapport par document ; aucun doublon silencieux
P1	Comparateur de fidélité	Affichage côte à côte PDF de référence / email bureau / email mobile, différences par bloc, validation humaine
P1	Bibliothèque de composants	En-têtes, pieds, colonnes, bandeaux, cartes, galerie ; composants MJML réutilisables et versionnés
P1	Conversion HTML avancée	Support des tableaux hérités, CSS inline, flex/grid approximés ; score de reconstruction fondé sur mesures explicites
P1	Compatibilité messagerie automatisée	Matrice clients/versions, captures de référence, détection des régressions typographiques et des éléments non pris en charge

P1	Contrôle éditorial et accessibilité	Ordre de lecture, descriptions d'images, contrastes, taille des liens et contenu textuel de remplacement
P2	Synchronisation éditoriale SPIP	Association blocs/articles/traductions ; prévisualisation des contenus réels avant publication
P2	Variantes multilingues	Traductions associées par identifiant, ajustement d'espaces, contrôle de débordements DE/IT/EN/FR
P2	Thèmes et chartes	Jetons de couleur, typographie de secours, marges, rayons, logos, compatibilité sombre/clair
P2	Plugin d'export pour InDesign	Commande ou script exportant un paquet de contenu structuré + manifeste JSON + ressources ; sans imposer un moteur MJML dans InDesign
P2	Extension QuarkXPress	Étudier capacités d'automatisation/API de la version présente avant de promettre une extension native ; exporter un paquet normalisé
P2	Historique collaboratif	Brouillons, approbation, différences entre versions, rôles et gel avant envoi
P3	Contrôle intelligent des blocs	Suggestions facultatives de remplacement des compositions non convertibles ; jamais de modification silencieuse
P3	Industrialisation et API interne	Traitement asynchrone sur serveur, limitation débit, tâches rejouables, métriques et webhook signé

24. Architecture V2 proposée

Trois modules découplés : (A) PLM-MJML-IMPORT pour InDesign, QuarkXPress et HTML ; (B) PLM-MJML-CORE pour parseur, normalisation JSON, générateur MJML, compilateur et audit ; (C) SPIP-MJML pour l'interface, les droits, les modèles, l'aperçu et Newsletters. Les fichiers sources et rendus sont conservés séparément. La base de données suit les règles canoniques PLM et ne doit pas être multipliée sans décision documentée.

Contrats internes : importer(source, profil) → paquet normalisé ; convertir(paquet, règles) → JSON + diagnostic ; rendre(JSON, cible) → MJML/HTML + diagnostic ; publier(version_approuvée) → squelette SPIP vérifié. Toutes les sorties transportent id source, numéro de version, date et empreinte SHA-256.

25. Critères d'acceptation quantifiés V2 et gouvernance

- Corpus de référence : au moins 10 exemples par origine (HTML, InDesign, QuarkXPress), en variant 1/2 colonnes, images, légendes, langues et tableaux ; taille à affiner lors de l'audit.
- Indicateurs : taux de conversion sans intervention, nombre de blocs non mappés, durée de conversion, nombre de liens invalides, taux d'images manquantes et erreurs par client de courriel.
- Comparaison visuelle : métrique d'image indicative uniquement ; la lisibilité mobile, la conservation du contenu et l'ordre de lecture prévalent sur le pixel à pixel.
- Sécurité : pas d'exécution de scripts issus d'un HTML importé, pas de chemins d'accès arbitraires, quotas de fichiers, protection contre les requêtes vers des adresses internes (SSRF, Server-Side Request Forgery).

- Validation : audit lecture seule → proposition dry-run → approbation explicite → exécution contrôlée → preuve et restauration vérifiable.
- Décision finale : ne pas inscrire « conforme production » tant que les versions SPIP, PHP (PHP: Hypertext Preprocessor), Node.js, QuarkXPress et InDesign n'ont pas été vérifiées sur la cible.

26. Sources et documentation complémentaire V002

MJML — documentation des composants : <https://documentation.mjml.io/>

MJML — compilateur et code : <https://github.com/mjmlio/mjml>

SPIP — développement et pipelines : <https://programmer.spip.net/>

Adobe — référence InDesign : <https://helpx.adobe.com/indesign/>

Quark — documentation produit : <https://www.quark.com/documentation/>

Can I Email — compatibilités CSS des courriels : <https://www.caniemail.com/>

W3C — accessibilité WCAG : <https://www.w3.org/WAI/standards-guidelines/wcag/>

Statut V002 : DOCUMENT MAÎTRE — PROPOSITION EN ATTENTE D'AUDIT CANONIQUE ; aucun dossier PLM modifié. Fin du document.

PARTIE II — V003 : ATELIER VISUEL MULTIFORMAT ET DOUBLE MOTEUR

Cette partie complète et, en cas de conflit, précise la spécification V002. L'ensemble des prescriptions précédentes reste présent pour traçabilité. Le « V003 » de ce document désigne la troisième révision documentaire ; les versions logicielles V0.x, V1 et V2 relèvent d'un calendrier distinct.

20. Décision structurante : un atelier, deux moteurs de publication

Nouvelle règle : ne pas considérer MJML comme l'unique format de sortie. Un import HTML (HyperText Markup Language) destiné aux courriels et jugé valide peut rester HTML. Un design structuré et responsive peut utiliser MJML (Mailjet Markup Language) puis être compilé en HTML. SPIP (Système de publication pour l'Internet) reste maître du contenu dynamique et de la publication.

Voie	Chaîne de transformation	Avantages	Inconvénients / conditions
HTML natif	HTML importé → assainissement → édition GrapesJS Newsletter → HTML de courriel → SPIP	Moins de transformations ; conserve certains tableaux de courriel existants	CSS (Cascading Style Sheets) arbitraire souvent incompatible ; édition non parfaitement réversible
MJML	Blocs PLM → GrapesJS MJML → MJML → compilation HTML → SPIP	Composants adaptés aux courriels ; meilleure cohérence responsive	Structures libres impossibles à représenter exactement ; conversion avec pertes
Hybride contrôlé	Import → modèle PLM → choix motivé HTML ou MJML → validation	Permet de choisir le meilleur export pour chaque maquette	Deux moteurs à tester, maintenir et comparer

21. Comparaison des éditeurs WYSIWYG

Éditeur	Capacités confirmées / visées	Avantages	Limites / statut
GrapesJS + preset-newsletter	Blocs de newsletter, import de template HTML, CSS inline à l'export	Open source ; centré HTML de courriel ; extensible	Import arbitraire ne préserve pas toutes les caractéristiques ; tests clients requis
GrapesJS + grapesjs-	Construction graphique	Conception structurée	Pas de fidélité absolue

mjml	par composants MJML ; aperçu de compilation	responsive ; composants mj-text, mj-image, mj-raw	aux compositions imprimées ; aperçu navigateur ≠ tous clients
Mosaico	Édition visuelle de newsletters sur modèles compatibles	Modèles simples et contraints	Intégration et réversibilité à étudier
Unlayer	Éditeur embarquable selon conditions commerciales	Expérience visuelle avancée	Licence, dépendance et format propriétaire à vérifier
Maizzle	Générateur de courriels HTML orienté développeurs	Outil de production/template, possible alternative au compilateur	Ne remplace pas un éditeur WYSIWYG

Choix V003 : GrapesJS comme éditeur principal, avec deux configurations distinctes dans une seule interface. Ne pas promettre un changement de moteur sans perte de données. Tout basculement génère une nouvelle révision et un rapport de différences.

22. Relation avec Adobe InDesign : chemin privilégié

36. Dans InDesign, préparer styles de paragraphe, styles de caractère, textes liés, ordre de lecture et objets d'images ; éviter la dépendance aux positions absolues.
37. Conserver la source native INDD (InDesign Document) sans modification. Exporter IDML (InDesign Markup Language) pour analyser les structures ; exporter aussi EPUB redistribuable (Electronic Publication), HTML lorsque disponible et PDF (Portable Document Format) comme témoin.
38. Importer séparément les ressources et noter chemins, polices sous licence, profils colorimétriques, variantes de mise en page et ordre de lecture.
39. Analyser IDML (paquet XML, Extensible Markup Language) ou XHTML (Extensible HyperText Markup Language) issu de EPUB ; associer styles à titres, paragraphes, légendes et blocs images.
40. Convertir les positions papier en sections/colonnes adaptatives ; produire un JSON (JavaScript Object Notation) intermédiaire avec provenance bloc par bloc.
41. Ouvrir le résultat dans GrapesJS, corriger les ruptures mobiles, marquer les éléments non convertibles et comparer avec PDF témoin.
42. Exporter HTML ou MJML suivant la structure ; prévisualiser et valider dans les principaux logiciels de messagerie avant de préparer un squelette SPIP.

Avantages : IDML donne une structure détaillée ; EPUB redistribuable fournit un ordre de lecture mieux adapté aux écrans. Inconvénients : coordonnées papier, habillages complexes, fonctions OpenType, transparences, compositions typographiques et images de fond ne se traduisent pas nécessairement en courriels éditables. Un EPUB à mise en page fixe est une référence visuelle, pas un pivot responsive universel.

23. Relation avec QuarkXPress : passerelle par publication numérique

43. Conserver le fichier QXP natif en lecture seule et déterminer la version de QuarkXPress et le type de projet.
44. Exporter une publication HTML5 et, si approprié, un EPUB redistribuable ; conserver un PDF de contrôle et les images liées.
45. Inventorier HTML, feuilles CSS, images, scripts et polices ; retirer les composants interactifs et les éléments non pris en charge par les courriels.
46. Reconstruire la hiérarchie éditoriale et l'ordre de lecture ; identifier les textes éventuellement vectorisés ou rasterisés.
47. Passer par le schéma JSON PLM puis par l'éditeur GrapesJS ; générer HTML de courriel ou MJML selon le diagnostic.
48. Mesurer les différences, conserver les corrections humaines et documenter les cas de sortie « image de remplacement ».

Avantages : QuarkXPress propose un export HTML5 et EPUB documenté. Inconvénients : HTML5 de publication est du HTML pour navigateur, pas nécessairement du HTML de courriel ; les scripts, animations, positionnements et interactions ne sont pas directement convertibles. La granularité des blocs dépend de la qualité de l'export et du document d'origine.

24. Import HTML : quatre canaux et leur fiabilité

Entrée	Opération	Contrôle nécessaire
Fichier .html	Lire le HTML et les fichiers associés localement	Encodage, CSS, chemins relatifs, sources de polices
Dossier ZIP	Extraire dans un bac à sable, réécrire les liens internes	Taille, traversée de chemin, liens symboliques, ressources externes
Adresse URL	Télécharger seulement des URL autorisées et ressources permises	SSRF (Server-Side Request Forgery : falsification de requêtes côté serveur), licences, authentification, suivi des redirections
Copier-coller HTML	Nettoyer et parser les fragments	Perte de styles, images externes, attributs dangereux

Procédure commune : import immuable → nettoyage de sécurité → analyse DOM (Document Object Model) et CSS → inférence de blocs → score de convertibilité → proposition HTML/MJML → édition → tests. Bannir toute exécution JavaScript de la source lors de l'analyse. Prévoir une quarantaine pour chaque ressource externe.

25. Contrat JSON PLM-EMAIL-DOC V002 et éditeur

- Pour chaque bloc : identifiant stable, type, ordre de lecture, texte, langue, styles compatibles courriel, médias, droits, source_file, source_locator, transform_history et flags de conversion.
- Préserver séparément document.source, document.edited, document.export_html et document.export_mjml ; ne pas prétendre que les exports sont des représentations sans perte.
- Ajouter export_strategy = html_native | mjml | image_fallback, justification et niveau de confiance ; validation humaine avant changement de moteur.
- Utiliser un schéma versionné, migrations testées, signatures SHA-256 (Secure Hash Algorithm, empreinte cryptographique de 256 bits) et stockage des originaux immuables.
- Distinguer les blocs dynamiques SPIP du HTML libre. Encoder les boucles SPIP comme composants structurés, avec placeholders contrôlés avant compilation MJML ; restaurer et valider après compilation.

26. Intégration SPIP : construction du squelette et limites

49. Lire les versions réelles de SPIP et des plugins Newsletters, MailSubscribers, MailShot et Facteur ; confirmer leur compatibilité avant de coder.
50. Définir les blocs SPIP autorisés : article, rubrique, sélection de contenus, logo, titre, texte, date, langue et URL absolue.
51. Créer une table de correspondance entre identifiant de bloc JSON et expression SPIP ; garder les expressions SPIP hors des zones éditables non fiables.
52. Générer un squelette de newsletter dans un répertoire de développement ; compiler MJML avant l'interprétation SPIP si cette voie a été choisie.
53. Restituer les directives SPIP et vérifier qu'aucune balise n'a été supprimée ni altérée ; tester sur un jeu de contenus connu.
54. Produire une version HTML statique de test, la version de production résolue par SPIP et une version texte accessible ; vérifier liens, images, consentements et désinscription.
55. Publier seulement après comparaison visuelle et approbation, avec export, manifeste de fichiers et retour arrière.

La voie exacte de connexion aux plugins SPIP doit être confirmée dans l'installation cible. Ne pas supposer que le seul renommage d'un HTML compilé en newsletter.html suffise dans toutes les configurations : modèles, contexte, critères d'envoi et hooks doivent être validés.

27. Fidélité graphique : règles de mesure et arbitrages

Critère	Mesure / test	Décision
---------	---------------	----------

Lisibilité	Titre, corps, contrastes, taille minimum sur mobile	Échec bloquant si contenu illisible
Ordre de lecture	Comparaison ordre éditorial source / sortie	Échec bloquant si narration déformée
Images	Ratio, recadrage, légende, texte alternatif	Accepter recadrage documenté
Typographie	Famille de repli, césure, interlignage	Accepter divergence documentée ; jamais garantir polices propriétaires
Responsive	Contrôle 320, 375, 600, 800 px, puis vrais clients	Échec bloquant si débordement essentiel
Clients de messagerie	Gmail, Apple Mail, familles Outlook pertinentes	Matrice de compatibilité versionnée
Accessibilité	Texte réel, descriptions alternatives, ordre et contrastes	Préférer reconstruction à rasterisation

Comparer systématiquement des captures à largeur donnée. Les métriques visuelles automatiques (différence d'image, similarité structurelle) orientent le contrôle mais ne remplacent pas l'inspection humaine : le rendu email dépend des clients, versions, modes sombre et chargement d'images.

28. Avantages, inconvénients et risques stratégiques

Choix	Avantage	Risque	Mesure
Deux moteurs HTML/MJML	Évite les conversions artificielles	Double maintenance et divergences	Tests identiques + matrice des capacités
GrapesJS intégré dans SPIP	Édition visuelle autonome	Surface d'attaque JavaScript et dépendances	Versions verrouillées, isolation et contrôle de rôles
Schéma JSON PLM	Traçabilité et multi-export	Évolution complexe du schéma	Migration contrôlée et contrats
Importer InDesign/QuarkXPress	Réemploi des maquettes professionnelles	Perte de la logique responsive	Guide d'export et contrôle humain
Images en repli	Fidélité visuelle rapide	Texte non éditable, accessibilité faible	Fallback limité et signalé
Services externes de conversion	Installation simplifiée	Confidentialité, coûts et disponibilité	Compilation locale par défaut si viable

29. Plan opératoire : prototype, pilote et production

56. AUDIT lecture seule : inventoirer PLM-MASTER, BIBLE canonique, README du dossier pressenti, versions et contraintes Infomaniak/SPIP. Restituer état sans créer de dossier.
57. JEU DE TEST : collecter au moins 3 newsletters HTML, 3 maquettes InDesign et 3 projets QuarkXPress de complexité croissante, avec exports disponibles et références PDF ; vérifier licences.
58. POC 1 : GrapesJS Newsletter autonome avec import/export HTML et sauvegarde de projet ; démontrer l'édition réversible sur un échantillon simple.
59. POC 2 : GrapesJS MJML autonome et compilation ; comparer les mêmes contenus à largeur mobile et bureau.
60. POC 3 : passerelles IDML/EPUB/HTML5 → schéma JSON ; produire un rapport par fichier sur les éléments perdus.
61. CHOIX : décider entre moteur HTML ou MJML au niveau de chaque modèle, sur résultats mesurés et non uniquement sur provenance.
62. PLUGIN : intégrer les éditeurs dans l'espace privé SPIP avec droits d'accès, validation, sauvegarde, historique et export uniquement.
63. PILOTE : brancher Newsletters dans un environnement de test ; envoyer uniquement à des boîtes de contrôle consentantes.

64. QUALITÉ : automatiser lint, schéma JSON, tests de sécurité, rendus de contrôle, gestion du versionnement et rollback.
65. GATE : présentation du dry-run ; approbation explicite avant toute installation ou altération des dossiers PLM de production.

30. Plan logiciel et estimation réaliste

Palier	Livrable principal	Charge indicative
P0 – Audit + preuves	Audit, corpus et critères d'acceptation	2–4 jours
P1 – HTML visuel	GrapesJS Newsletter, import local, aperçu, export	5–8 jours
P2 – Double moteur	MJML, choix de moteur, JSON et traçabilité	7–12 jours
P3 – Connecteurs	InDesign IDML/EPUB + QuarkXPress HTML5/EPUB	8–15 jours
P4 – SPIP	Plugin, modèles, balises, contrôle et prévisualisation	6–10 jours
P5 – Assurance qualité	Tests clients, sécurité, doc, migration/rollback	5–10 jours
TOTAL V003 élargie	1 développeur, maquettes structurées, outils existants	33–59 jours effectifs

Estimation : 33–59 jours de développement, plus 20 à 30 % de réserve selon les fichiers réellement testés. La précédente estimation de 20–35 jours concernait un périmètre plus restreint ; 14–25 jours additionnels avaient été envisagés pour l'éditeur et l'import avancé. Les durées ne sont pas des résultats de benchmark et n'incluent ni licences commerciales, ni infrastructure, ni délais de décision.

31. Séquence des versions de logiciel proposées

Version	Critère d'acceptation
V0.1	HTML importé, rendu, modifié et exporté sans perte de contenu critique
V0.2	MJML créé visuellement, compilé, importé/exporté ; test mobile
V0.3	Schéma JSON V002 et historique de provenance/reconstruction
V0.4	Import URL, ZIP et sanitisation renforcée
V0.5	Passerelle InDesign testée sur trois fichiers
V0.6	Passerelle QuarkXPress testée sur trois fichiers
V0.7	Génération de squelette SPIP, contenu dynamique protégé
V0.8	Contrôle de fidélité visuelle et clients courriel
V0.9	Tests intégrés, journal, rollback, documentation et revue de sécurité
V1.0	Pilote approuvé et mis en production sur périmètre limité
V2.0	Import batch, bibliothèque réutilisable, workflow d'approbation et métriques

32. Perspectives V2.0 : fonctions avancées

- Bibliothèque de composants partagés : en-têtes, pieds de page, photographies, notices et styles institutionnels, avec versionnement et compatibilités par moteur.
- Import en lots et files d'attente ; rapports d'échec par document, reprises idempotentes et limites de ressources.
- Comparateur visuel original/export avec superposition, différences et validation de chaque bloc sensible.
- Assistants de migration : reconnaître les styles répétés depuis InDesign/QuarkXPress et proposer des composants, sans modifications silencieuses.
- Variantes multilingues français/allemand/italien/anglais, avec validation des débordements et contenus traduits.
- Workflow éditorial : brouillon, révision, validation, gel de version, tests d'envoi, publication, conservation de preuve.
- Contrôle des licences des polices, photographies et textes ; export audit de provenance et des consentements.
- API (Application Programming Interface) interne versionnée pour importer/exporter, sans exposition publique initiale.
- Télémétrie locale minimisée : taux d'échec, blocs non convertibles, corrections manuelles, matrice des clients et qualité perçue.
- Comparaison entre moteurs : proposer HTML natif quand MJML dégrade la source, et MJML lorsque le résultat responsive est supérieur.

33. Registre de preuves, décisions et gouvernance canonique

Chaque sortie livrée devra enregistrer : source et licence, checksum SHA-256, version moteur, versions GrapesJS/MJML, date, schéma JSON, stratégie d'export, auteur de validation, rapport de différences, résultat des tests email et rollback_target. Toute modification dans PLM exige : (1) constat réel ; (2) audit lecture seule ; (3) proposition avec dry-run ; (4) autorisation explicite ; (5) exécution et preuve ; (6) plan de retour arrière. La destination /Users/adam/A_PHOTO-LAB-MASTER/PLM-MJML/ est proposée et NON confirmée.

34. Sources et documentation technique — accès officiel

GrapesJS documentation — <https://grapesjs.com/docs/>

GrapesJS Newsletter — <https://github.com/GrapesJS/preset-newsletter>

GrapesJS MJML — <https://github.com/GrapesJS/mjml>

MJML documentation — <https://documentation.mjml.io/>

Adobe InDesign : export EPUB — <https://helpx.adobe.com/indesign/desktop/save-export-and-publish/export-to-epub/export-to-epub.html>

QuarkXPress : Digital Publishing 2026 —

https://www.quark.com/documentation/quarkxpress/2026/english/Digital%20Publishing%20with%20QuarkXPress%202026/_7bb86344-83c1-4d6b-8e9b-8e11743d70ca.html

SPiP Newsletters — <https://plugins.spip.net/newsletters.html>

SPiP Mailshot — <https://plugins.spip.net/mailshot.html>

Documentation développeur SPiP — <https://programmer.spip.net/>

FIN DE V003 — Document de conception ; aucun test réel sur les fichiers personnels, aucune installation, aucun déploiement ou modification de l'arborescence PHOTO LAB MASTER.

PARTIE III — V004 : COMPARATIFS, PASSERELLES PAO ET INSTALLATION DES DEUX ÉDITEURS

Édition : 8 octobre 2026. Statut : spécification consolidée, installation non effectuée. Les parties précédentes V001 à V003 sont conservées. Ce chapitre V004 prévaut en cas de divergence architecturale.

35. Clarification fondamentale : InDesign et QuarkXPress ne disparaissent pas

Adobe InDesign et QuarkXPress sont les outils auteurs de publication assistée par ordinateur (PAO). GrapesJS n'ouvre pas directement leurs formats natifs .indd ou .qxp. Une passerelle d'export et de normalisation est nécessaire ; l'objectif est de préserver les textes, les images et les structures éditables, pas de promettre une reproduction pixel à pixel dans tous les clients de messagerie.

Origine	Export/entrée préférée	Conversion PLM	Destination
InDesign	HTML ou EPUB redistribuable ; IDML pour extraction avancée	Décodage HTML/EPUB → composants de courriel	GrapesJS Newsletter ; puis SPIP
QuarkXPress	HTML5 Publication ou EPUB redistribuable si disponible	Extraction et normalisation du HTML, images et styles	GrapesJS Newsletter ; puis SPIP
HTML déjà existant	HTML + ressources ou archive ZIP	Assainissement, liens absolus, styles et blocs	GrapesJS Newsletter ; puis SPIP
Nouvelle lettre responsive	MJML natif	Composants MJML visuels ; compilation HTML	GrapesJS + MJML ; puis SPIP
PDF/EPS de référence	PDF (Portable Document Format) / EPS (Encapsulated PostScript)	Référence visuelle, pas format de structure principal	Contrôle de fidélité seulement

36. Comparaison détaillée des deux solutions retenues

Dimension	GrapesJS Preset Newsletter	GrapesJS MJML
Modèle éditable	Projet GrapesJS (JSON interne) + HTML/CSS	Projet GrapesJS (JSON interne) + MJML
Saisie principale	HTML existant ou blocs email	Composants mj-section, mj-column, mj-text...
Import HTML issu de PAO	Prioritaire, exige nettoyage	Non direct ; reconstruire en MJML
Fidélité aux compositions	Plus flexible, mais forte dépendance aux tableaux et clients	Plus contraint ; réorganisation responsive
Responsive	À construire, éprouver sur plusieurs clients	Comportements prévus par compilateur MJML, contrôle nécessaire
Exports	HTML avec styles intégrés (preset)	MJML ; HTML après compilation contrôlée
Code source réversible	Non pour HTML arbitraire	Non pour HTML arbitraire ; oui pour projet MJML sauvegardé
Lien SPIP	Transformer HTML final en squelette validé	Compiler MJML avant interprétation des balises SPIP
Maintenance	CSS de courriel et composants	Versions du compilateur MJML et

	HTML spécifiques	composants
Meilleur cas	Reprendre une newsletter HTML, InDesign ou QuarkXPress	Créer de zéro un gabarit responsive
Risque majeur	HTML importé devient mal éditable ou cassé dans Outlook	Perte de liberté graphique et conversion HTML→MJML non fidèle

37. Variantes de conversion : avantages, limites et décision

Variante	Avantages	Inconvénients	Décision
A — PAO → HTML → Newsletter → SPIP	Évite conversion MJML ; récupération possible des contenus	Styles impression/web incompatibles email ; corrections	Voie prioritaire PAO
B — PAO → HTML → modèle JSON → MJML → SPIP	Composants propres et homogènes, mobile	Reconstruction coûteuse, perte de positionnement libre	Option pour modèles réguliers
C — PAO → IDML/EPUB → moteur sémantique → double sortie	Meilleure analyse des styles et ordre de lecture	Parser et règles métier à développer ; fragilité aux versions	Recherche / phase avancée
D — PDF/EPS → rendu image → newsletter	Visuellement proche à largeur donnée	Texte non éditable, poids, accessibilité, mobile dégradés	Seulement bloc exceptionnel
E — Dessin direct dans MJML	Rapide pour nouveaux modèles, compilation standard	Pas de transfert direct fidèle PAO	Voie prioritaire création
F — HTML existant → SPIP sans MJML	Moins de transformations et dépendances	Nettoyage et tests CSS indispensables	Voie prioritaire archives

38. Schéma de travail et conservation des sources

Chaque import produit une liasse traçable : 01_SOURCE_ORIGINALE (fichiers auteur), 02_EXPORT_PAO (HTML/EPUB/IDML, images), 03_NORMALISATION (HTML assaini), 04_PROJET_EDITEUR (JSON GrapesJS et manifeste), 05_PUBLICATION (MJML éventuel, HTML final, squelette SPIP), 06_QA (captures et tests), 07_PREUVES (hash SHA-256, versions, décisions et droits). Ce schéma est PROPOSÉ et ne crée aucun dossier.

Règle : ne jamais considérer le HTML rendu comme une copie complète du projet visuel. Garder aussi l'état JSON GrapesJS, la version exacte des dépendances et le document PAO original.

39. Installation A — GrapesJS Preset Newsletter : procédure pas à pas

Cible : poste de développement local (iMac Intel ou Mac Apple Silicon) ; installation d'essai hors de la racine canonique PLM tant que le précontrôle n'a pas été validé. Ne pas installer directement dans le site SPIP de production.

- A0 — Lecture seule : vérifier versions macOS, node --version, npm --version, git --version, version de SPIP, configuration serveur, README PLM-MASTER et README du futur dossier PLM-MJML ; relever droits et sauvegardes.
- A1 — Préparer un répertoire temporaire de test explicitement autorisé (non canonique), puis initialiser un projet Vite (outil de compilation frontend) ; aucune modification de SPIP.
- A2 — Installer GrapesJS et le paquet officiel grapesjs-preset-newsletter, versions verrouillées dans package-lock.json.
- A3 — Créer index.html et src/main.js ; importer grapesjs/dist/css/grapes.min.css ; initialiser l'éditeur Newsletter.
- A4 — Démarrer serveur local, vérifier blocs, insertion d'images, importation HTML et commande d'export HTML avec styles intégrés.

- A5 — Tester trois HTML : newsletter simple, HTML export InDesign et HTML5 QuarkXPress normalisé ; conserver les divergences.
- A6 — Mesurer qualité écran étroit/grand écran, liens, poids, images, tableaux, Outlook/Gmail/Apple Mail ; archiver capture et résultats.
- A7 — Après validation, fabriquer des fichiers front-end statiques pour SPIP ; encapsuler dans une page privée SPIP avec autorisations explicites et protection contre les scripts importés.

Commandes indicatives pour un répertoire de laboratoire autorisé :

```
npm create vite@latest plm-mjml-newsletter-test -- --template vanilla
cd plm-mjml-newsletter-test
npm install
npm install grapesjs grapesjs-preset-newsletter
npm run dev
```

Fichier src/main.js minimal :

```
import grapesjs from 'grapesjs';
import 'grapesjs/dist/css/grapes.min.css';
import newsletter from 'grapesjs-preset-newsletter';
const editor = grapesjs.init({
  container: '#app', height: '85vh', storageManager: false,
  plugins: [newsletter],
});
// Export HTML email via commande du preset (contrôler API/retour selon version)
// editor.runCommand('gjs-get-inlined-html');
```

Fichier index.html : prévoir <div id="app"></div> et <script type="module" src="/src/main.js"></script>. Ce fragment démontre le chargement ; stockage, authentification et intégration SPIP restent à programmer.

40. Installation B — GrapesJS + MJML : procédure pas à pas

- B0 — Même audit lecture seule ; isoler ce banc d'essai du preset Newsletter pour limiter les interférences de composants.
- B1 — Initialiser un deuxième projet Vite ; installer grapesjs, grapesjs-mjml et mjml pour la compilation finale contrôlée.
- B2 — Ajouter un conteneur #app et charger le plugin MJML dans GrapesJS.
- B3 — Créer un premier modèle avec mjml, mj-body, mj-section, mj-column et mj-text ; contrôler édition, export du document MJML et prévisualisation.
- B4 — Compiler le MJML final avec le compilateur officiel hors navigateur ; comparer la sortie HTML à l'aperçu de l'éditeur.
- B5 — Injecter des marqueurs de substitution sécurisés, puis tester la restauration des boucles SPIP avant publication.
- B6 — Conserver sources MJML, JSON du projet, HTML compilé, résultats de tests ; valider avant déploiement SPIP.

```
npm create vite@latest plm-mjml-editor-test -- --template vanilla
cd plm-mjml-editor-test
npm install
npm install grapesjs grapesjs-mjml mjml
npm run dev
```

Fichier src/main.js minimal :

```
import grapesjs from 'grapesjs';
import 'grapesjs/dist/css/grapes.min.css';
import mjmlPlugin from 'grapesjs-mjml';
const editor = grapesjs.init({
  container: '#app', height: '85vh', storageManager: false,
  plugins: [mjmlPlugin],
});
```

Compilation d'un document MJML exporté :

```
npx mjml newsletter.mjml -o newsletter.html
```

Attention : les extraits sont des amorces de prototype, pas un installateur PLM ni un plugin SPIP prêt pour la production. Les versions npm doivent être figées puis soumises à l'audit de licences et de vulnérabilités. Ne pas lancer npm audit fix --force sans validation.

41. Procédure complète InDesign → SPIP : deux parcours

Parcours privilégié : (1) dupliquer le document InDesign ; (2) appliquer styles de paragraphe et ordre de lecture ; (3) exporter HTML si disponible dans la version utilisée ou EPUB redistribuable ; (4) inspecter ressources et textes ; (5) normaliser HTML/CSS, supprimer scripts et interactions ; (6) convertir en tableaux/blocs compatibles email ; (7) importer GrapesJS Newsletter ; (8) corriger colonnes et médias ; (9) insérer emplacements SPIP dans les composants autorisés ; (10) générer HTML final et tester ; (11) publier en squelette de newsletter après validation.

Parcours avancé : exporter IDML (InDesign Markup Language), extraire références de pages/blocs/styles/ordre de lecture, reconstruire un JSON éditorial puis choisir la sortie HTML ou MJML. Le format IDML ne donne pas directement du MJML ; il nécessite le développement d'un interpréteur.

Pour un EPUB redistribuable, la source s'apparente à des fichiers XHTML (Extensible HyperText Markup Language) et ressources compressées ; ne pas supposer que le livre électronique est déjà un email compatible.

42. Procédure complète QuarkXPress → SPIP : deux parcours

Parcours HTML5 : (1) vérifier version QuarkXPress et modèle ; (2) exporter « File > Export as > HTML5 Publication » suivant documentation ; (3) inventorier HTML, ressources, scripts et mises en page ; (4) extraire uniquement la structure éditoriale nécessaire ; (5) supprimer JavaScript, animations et positions absolues non prises en charge ; (6) normaliser CSS et images ; (7) importer GrapesJS Newsletter ; (8) reconstruire les blocs pour les courriels ; (9) prévisualiser et effectuer des envois de test ; (10) publier via SPIP.

Parcours EPUB : privilégier une sortie redistribuable si disponible ; récupérer les XHTML/ressources de l'archive EPUB ; reconstruire l'ordre de lecture ; vérifier les images et textes avant édition GrapesJS. Une publication HTML5 QuarkXPress vise d'abord le navigateur et non les logiciels de courriel.

43. Intégration dans SPIP : sécurité et séparation des rôles

Composants recommandés : backend PHP de SPIP responsable des droits, stockage et audit ; éditeur GrapesJS servi sous forme de ressources statiques validées ; service de conversion contrôlé ; Newsletters pour composition, Facteur pour transport, Mailshot selon dispositif d'envoi. Ne jamais considérer le résultat HTML importé comme sûr : assainir HTML, CSS et URL, refuser scripts, iframes et accès locaux ; imposer une politique de ressources médias.

Les boucles SPIP ont une syntaxe différente du HTML et du MJML. Ne pas les insérer aveuglément dans un parseur WYSIWYG : préférer des composants « donnée SPIP » de l'éditeur, exportés en marqueurs protégés, puis restaurés et testés côté serveur. Interdire les valeurs arbitraires non autorisées dans les conditions de boucle. Vérifier envoi d'essai, modèle de désabonnement et liens absolus.

44. Matrice de validation et choix de production

Test de réception	Newsletter	MJML	Condition d'acceptation
Import HTML PAO simple	Obligatoire	Optionnel	Texte complet, images et liens intacts
Import HTML PAO complexe	Obligatoire	Étude	Défauts documentés et correction possible
Création de zéro responsive	Utile	Obligatoire	Mobile/desktop et tableaux de contenu validés
Export et réimport projet	Obligatoire	Obligatoire	Conservation du JSON projet et du code source
Balises SPIP	Obligatoire	Obligatoire	Aucune boucle perdue ni interprétée précocement
Rendu email	Obligatoire	Obligatoire	Gmail, Apple Mail, Outlook, images

			désactivées
Sécurité	Obligatoire	Obligatoire	Aucun script exécutable importé ; autorisations vérifiées
Traçabilité	Obligatoire	Obligatoire	Versions, SHA-256, journal des conversions

45. Charge et jalons actualisés V004

Lot	Charge indicative (jours ouvrés)	Dépendance
Précontrôle BIBLE/README, audit lecture seule, corpus	2-4	Accès versions et échantillons réels
Banc GrapesJS Newsletter + imports HTML	4-7	Export PAO testable
Banc GrapesJS MJML + compilation	3-5	Node.js et versions figées
Normaliseur HTML PAO + ressources	5-9	Jeu d'essai et critères
Plugin SPIP, marqueurs dynamiques, stockage	7-12	Environnement SPIP de test
Passerelle avancée IDML/EPUB	5-10	Fichiers représentatifs
Tests messagerie, sécurité, documentation	5-9	Banc de test et validation humaine

Ordre de grandeur V004 : 31-56 jours ouvrés pour les lots ci-dessus, hors extension avancée de fidélité pixel et hors travail déjà réalisé. Ajouter 20 à 30 % de réserve en cas de maquettes complexes. Cette fourchette remplace les comparaisons d'effort précédentes pour le périmètre V004 ; elle n'est pas un devis.

46. Programme de livraison et versions

V0.1 audit + échantillons ; V0.2 éditeur Newsletter autonome ; V0.3 éditeur MJML autonome ; V0.4 normaliseur HTML ; V0.5 premier import InDesign ; V0.6 premier import QuarkXPress ; V0.7 plugin SPIP de test ; V0.8 sécurité et squelettes dynamiques ; V0.9 tests de courriels et recette ; V1.0 production contrôlée ; V2.0 conversion par lots, bibliothèque de composants, comparaison automatique, correspondance sémantique PAO, versions, langues et tableau de bord de qualité.

47. Gouvernance PLM et conditions avant application

Avant changement réel sur iMac, Mac M5, serveur ou base de données PLM : 1) lire la dernière BIBLE dans PLM-MASTER et les README concernés ; 2) vérifier l'état réel ; 3) auditer en lecture seule ; 4) préparer un dry-run avec destination, dépendances, diff, SHA-256, sauvegarde et ordre de rollback ; 5) obtenir validation explicite ; 6) seulement alors exécuter. La destination A_PHOTO-LAB-MASTER/PLM-MJML reste PROPOSÉE et non vérifiée sur la machine.

48. Sources primaires et liens de téléchargement / documentation

- GrapesJS Newsletter — dépôt officiel — <https://github.com/GrapesJS/preset-newsletter>
- GrapesJS MJML — dépôt officiel — <https://github.com/GrapesJS/mjml>
- GrapesJS — documentation — <https://grapesjs.com/docs/>
- MJML — documentation — <https://documentation.mjml.io/>
- Adobe — exporter en EPUB depuis InDesign — <https://helpx.adobe.com/indesign/desktop/save-export-and-publish/export-to-epub/export-to-epub.html>
- Adobe — options EPUB — <https://helpx.adobe.com/indesign/desktop/save-export-and-publish/export-to-epub/epub-export-options.html>

- QuarkXPress 2026 — publication numérique —
https://www.quark.com/documentation/quarkxpress/2026/english/Digital%20Publishing%20with%20QuarkXPress%202026/_7bb86344-83c1-4d6b-8e9b-8e11743d70ca.html
- SPIP — programmer — <https://programmer.spip.net/>
- Vite — démarrage — <https://vite.dev/guide/>
- Node.js — téléchargements — <https://nodejs.org/en/download>

Remarque : les procédures externes varient selon la version exacte des logiciels, leur licence et la configuration du poste. L'adaptation aux clients email doit être testée ; aucun outil cité ne garantit une conversion WYSIWYG parfaite depuis la PAO.

49. Installation de GrapesJS sur macOS — principe et compatibilité

GrapesJS est une bibliothèque JavaScript (langage de programmation exécuté notamment dans le navigateur), non une application macOS autonome à installer comme Adobe InDesign. Deux modalités distinctes : (A) consulter un éditeur hébergé via navigateur ; (B) exécuter localement une application de développement contenant GrapesJS avec Node.js (environnement d'exécution JavaScript) et npm (Node Package Manager, gestionnaire de paquets). Ne jamais confondre compatibilité du navigateur et compatibilité des outils locaux.

Versions macOS : le projet Node.js documente pour les binaires actuels une cible macOS 13.5 ou supérieure, en Intel x86-64 et Apple Silicon ARM64. Sur Intel, le niveau de support exact varie suivant la branche Node.js ; vérifier la branche réellement installée. Ce seuil décrit Node.js moderne et NON GrapesJS affiché depuis un serveur. Pour macOS 10.15 Catalina, 11 Big Sur ou 12 Monterey, ne pas installer par défaut une branche récente de Node.js : privilégier un éditeur hébergé sur machine compatible ou serveur dédié et consulter sa compatibilité navigateur. Des versions Node.js historiques peuvent fonctionner sur d'anciens macOS, mais sont potentiellement hors maintenance et ne conviennent pas à une mise en production connectée à Internet.

Sur iMac Intel ancien : commencer par un audit en lecture seule de la version macOS, architecture processeur, navigateur, Node.js, npm et disponibilité de ports. Sur Mac Apple Silicon : utiliser la version ARM64 adaptée à macOS, sous réserve du système installé. Ne jamais modifier l'environnement PLM ni les répertoires canoniques sans validation explicite.

49.1 Matrice de compatibilité à appliquer

Mac / système	Affichage d'un GrapesJS hébergé	Développement local actuel
macOS 13.5 ou plus récent, Intel		
	Oui avec navigateur moderne pris en charge	
		Node.js moderne possible ; support Intel dépend de la branche
macOS 13.5 ou plus récent, Apple Silicon		
	Oui avec navigateur moderne pris en charge	
		Node.js moderne ARM64 : voie privilégiée
macOS 12 Monterey ou antérieur		
	Conditionné à la version et au moteur du navigateur	
		Pas de garantie avec Node.js actuel ; éviter tout forçage
Infomaniak / serveur web		
	Accès via navigateur à une interface déployée	

		Vérifier préalablement les capacités de build, le déploiement statique et la politique d'hébergement
--	--	--

49.2 Audit préalable en lecture seule (macOS)

Ouvrir Terminal et exécuter successivement les commandes suivantes ; elles ne modifient aucun fichier :

```
sw_vers -productVersion
```

```
uname -m
```

```
command -v node && node --version
```

```
command -v npm && npm --version
```

```
command -v git && git --version
```

Consigner macOS, architecture, versions trouvées et machine concernée. L'absence de Node.js n'empêche pas l'utilisation d'un éditeur GrapesJS déjà hébergé. Aucun « sudo », aucun installateur automatique et aucune création de dossier PLM à ce stade.

50. Installation locale — variante A : GrapesJS Newsletter (HTML direct)

Usage : importer et corriger le HTML (HyperText Markup Language) issu d'InDesign/QuarkXPress après normalisation ; produire du HTML de courriel et des styles CSS (Cascading Style Sheets) intégrés. L'import n'assure pas la fidélité graphique parfaite ni la compatibilité de tous les clients de messagerie.

- Prérequis : macOS compatible avec la version Node.js retenue ; installation officielle de Node.js et npm après validation de l'audit ; navigateur moderne ; réseau uniquement pour téléchargement initial ; emplacement de prototype non canonique approuvé.
- 1. Créer un projet JavaScript de test isolé, hors production PLM ; enregistrer le manifeste des dépendances et un fichier de verrouillage pour figer leurs versions.
- 2. Dans ce projet : `npm init -y`
- 3. Installer l'éditeur et le prérequis : `npm install grapesjs grapesjs-preset-newsletter`
- 4. Installer un serveur de développement adapté : `npm install --save-dev vite`
- 5. Créer `index.html`, `src/main.js` et importer « `grapesjs/dist/css/grapes.min.css` », « `grapesjs` », « `grapesjs-preset-newsletter` ». Initialiser `grapesjs.init({ container: "#gjs", height: "85vh", storageManager: false, plugins: [newsletterPlugin] })`.
- 6. Démarrer le prototype via `npx vite --host 127.0.0.1` puis ouvrir l'adresse locale affichée. Ne pas exposer le serveur de développement sur le réseau.
- 7. Importer d'abord un HTML simple ; valider l'édition des textes, des blocs et la sauvegarde du projet JSON (JavaScript Object Notation) ; générer ensuite le HTML avec CSS intégré.
- 8. Vérifier visuellement dans trois clients de messagerie et contrôler les images, les liens absolus, les tables, les textes alternatifs et les media queries (règles CSS conditionnelles).
- 9. Dans SPIP (Système de publication pour l'Internet), intégrer seulement le HTML validé dans un squelette autorisé ; protéger les balises dynamiques SPIP et effectuer un envoi de test.

51. Installation locale — variante B : GrapesJS + MJML (responsive)

Usage : construire une newsletter à partir de composants MJML (Mailjet Markup Language), puis compiler le MJML en HTML final. Adaptation aux petits écrans facilitée mais non garantie sans tests réels ; les imports HTML d'InDesign/QuarkXPress exigent une reconstitution structurée.

- Prérequis identiques à la variante A ; créer un prototype séparé, isolé des installations PLM actives.
- 1. Initialiser un projet de test isolé : `npm init -y`
- 2. Installer les bibliothèques : `npm install grapesjs grapesjs-mjml mjml`
- 3. Installer Vite pour le prototype : `npm install --save-dev vite`
- 4. Dans `src/main.js`, importer GrapesJS, `grapesjs-mjml` et la feuille CSS de GrapesJS ; initialiser `grapesjs.init({ container: "#gjs", height: "85vh", storageManager: false, plugins: [mjmlPlugin] })`.

- 5. Démarrer npx vite --host 127.0.0.1 ; créer une section, une colonne, un titre, une image et un bouton avec l'éditeur visuel.
- 6. Exporter et enregistrer le MJML source ; compiler localement un fichier newsletter.mjml avec : npx mjml newsletter.mjml -o newsletter.html
- 7. Comparer code MJML, HTML compilé, prévisualisation navigateur et affichage des courriels réels ; noter les écarts de colonnes, typographies et images.
- 8. Protéger les directives SPIP avant compilation ; éviter d'insérer des boucles brutes dans les zones MJML sans mécanisme testé ; après compilation, valider leur restitution et exécution côté SPIP.
- 9. Archiver projet GrapesJS, MJML, HTML de sortie, captures de tests et empreintes Secure Hash Algorithm 256-bit (SHA-256) dans le circuit de validation PLM.

52. Comparaison décisionnelle installation et chaîne d'import

Critère	A — GrapesJS Newsletter	B — GrapesJS + MJML
Entrée préférable		
	HTML exporté/nettoyé depuis InDesign/QuarkXPress	
		Création directe de composants responsives
Compilation MJML nécessaire		
	Non	
		Oui
Import HTML existant		
	Oui, avec nettoyage et reconstruction partielle	
		Pas de conversion HTML → MJML native universelle
Responsive		
	À construire, inspecter et tester	
		Facilité par les composants ; tester tout de même
Fidélité maquette print		
	Possiblement mieux conservée si HTML exploitable	
		Souvent plus de restructuration
Installation macOS		
	Même exigence Node.js si prototype local	
		Même exigence Node.js si prototype local
Sortie vers SPIP		
	HTML adapté au squelette	
		MJML compilé en HTML, puis adapté au squelette

53. Flux opérationnels et décision responsive / non responsive

FLUX A : QuarkXPress ou InDesign → export HTML/EPUB/IDML selon possibilités → extraction et normalisation HTML → GrapesJS Newsletter → contrôles ordinateur/mobile → HTML de courriel compatible → squelette SPIP → test d'envoi. MJML n'est PAS une étape obligatoire.

FLUX B : maquette source ou nouvelle composition → analyse des blocs → reconstruction MJML dans GrapesJS + MJML → compilation MJML vers HTML → contrôles ordinateur/mobile → squelette SPIP → test d'envoi. Le HTML n'est pas une étape d'édition intermédiaire obligatoire, mais il est la sortie finale du compilateur.

FLUX C : newsletter HTML déjà conforme → audit HTML/CSS/images/liens et accessibilité → adaptation SPIP → tests et envoi. GrapesJS et MJML sont facultatifs.

Responsive ≠ rendu identique : privilégier colonnes empilables, largeur fluide sous conteneur de 600 px à titre de convention et non de garantie, images redimensionnables, texte lisible et ordre de lecture contrôlé ; produire si nécessaire une variante fixe de référence et une variante adaptative. Tester Gmail, Apple Mail et Outlook sur supports ciblés.

54. Conditions de validation avant installation PLM

- Relire la BIBLE PHOTO LAB MASTER active et les README de tous les dossiers affectés ; établir une photographie en lecture seule de l'existant.
- Préparer un dry-run contenant destinations, dépendances, ports, droits, chemins, fichiers créés, plan de retour arrière et résultats attendus.
- Ne lancer l'installation réelle, la création d'un dossier canonique ou le déploiement SPIP qu'après accord explicite.
- Séparer données de test, sources originales, fichiers éditables, compilations, tests de rendu et éléments de preuve.
- Conserver la V004 comme seul numéro de version du présent fichier ; ne pas publier une variante V004-R supplémentaire.

55. Références officielles complémentaires — installation et compatibilité

- GrapesJS — introduction, téléchargement et modes d'intégration : <https://grapesjs.com/docs/>
- GrapesJS — démarrage JavaScript et import des feuilles de style : <https://grapesjs.com/docs/getting-started.html>
- GrapesJS Newsletter — installation npm et options : <https://github.com/GrapesJS/preset-newsletter>
- GrapesJS MJML — dépôt officiel : <https://github.com/GrapesJS/mjml>
- Node.js — plateformes, versions minimales et classes de support : <https://github.com/nodejs/node/blob/main/BUILDING.md>
- Node.js — téléchargements officiels : <https://nodejs.org/en/download>
- Vite — guide officiel : <https://vite.dev/guide/>
- MJML — documentation et compilation : <https://documentation.mjml.io/>