## **Journal de bord du stage**

### **17 mars 2025 - Préparation du projet (avant le codage)**

**Tâches réalisées :**

* Définition du périmètre du projet, du workflow et des exigences fonctionnelles.
* Conception du schéma de base de données (tables : annonces, media, catégories).
* Mise en place de la structure du projet (répertoires, organisation des fichiers).
* Planification des interactions entre les contrôleurs, modèles et méthodes.
* Confirmation du traitement des fichiers médias (stockage temporaire avant soumission).
* Discussion sur les règles de validation et le parcours utilisateur.

**Interactions :**

* Participation à une réunion de présentation d'équipe (9h à 11h) où j'ai rencontré l'équipe.
* Réception de la description détaillée du projet par Guillaume.
* Pose de nombreuses questions à Guillaume pour clarifier certains aspects du projet.
* Prise de notes détaillées publiées sur ma page Notion : [Projet Integration II](https://www.notion.so/Projet-Integration-II-1b9049a845d4806d85a2c89418cf94d0?pvs=4)

**Problèmes rencontrés :**

* Aucun problème rencontré.

### **18 mars 2025 - Implémentation technique**

**Tâches réalisées :**

* Mise en place de la connexion à la base de données (db.php) et validation des requêtes.
* Création des fonctionnalités CRUD pour :
  + Table annonces (gestion des annonces).
  + Table media (liée aux annonces).
  + Table catégories (gestion des catégories).
* Développement du modèle générique (Model.php) comme classe parente pour la gestion des requêtes SQL.
* Implémentation des modèles spécifiques : annonce.php, media.php et categories.php.
* Tests complets des opérations CRUD sur chaque entité (insertion, récupération, mise à jour, suppression).
* Vérification approfondie de l'intégrité relationnelle entre les médias et les annonces.
* Finalisation de la structure globale du projet (contrôleurs, modèles et vues).
* Début de la préparation pour l'intégration de l'interface utilisateur (affichage des annonces et soumission des formulaires).

**Interactions :**

* Début de l’implémentation du backend (connexion base de données, CRUD).
* Échanges ponctuels avec Guillaume pour des clarifications techniques et fonctionnelles sur certaines caractéristiques du projet.
* Début des tests préliminaires sur le frontend en fin de journée.

**Problèmes rencontrés :**

* Quelques difficultés à comprendre clairement certains détails des fonctionnalités demandées.

**19 mars 2025 - Correction de bugs et validation des formulaires**

**Tâches réalisées :**

* Résolution d'un problème lié à Mustache causé par une balise </head> manquante.
* Débogage de l'affichage des catégories dans le formulaire de soumission.
* Correction d'un problème empêchant l'insertion des annonces dans la base de données.
* Ajout de la validation des numéros de téléphone au format international avec intl-tel-input.
* Implémentation d'un système permettant d'afficher/masquer un exemple de numéro de téléphone pour garder le formulaire propre.
* Ajout de logs pour suivre l'exécution des requêtes SQL et déboguer les erreurs d'insertion.
* Tests et ajustements sur l'interface utilisateur pour améliorer l'affichage du formulaire.

**Interactions :**

* Aucune interaction significative avec l'équipe, à part quelques échanges légers.
* La majorité de l'équipe travaillait à distance tandis que je travaillais sur place.

**Problèmes rencontrés :**

* Bug lié à Mustache empêchant l'affichage des catégories, résolu après avoir identifié l'erreur de syntaxe HTML.
* Requêtes d'insertion qui échouaient silencieusement en raison d'un problème avec le champ telephone, qui nécessitait un format numérique valide.
* Complexité de la gestion des numéros de téléphone avec validation internationale.
* Difficulté à maintenir la propreté du formulaire tout en affichant les exemples de formats requis.

**20 mars 2025 - Intégration et validation des formulaires**

**Tâches réalisées :**

* Travail intensif sur l'intégration des formulaires avec Mustache.
* Résolution de plusieurs problèmes liés aux validations de formulaires.
* Ajustements et corrections sur l'affichage des erreurs dans les champs de saisie.
* Débogage des incohérences entre les fichiers Mustache et le système de validation.
* Test et validation des soumissions de formulaires après corrections.
* Vérification de la gestion des erreurs côté client et côté serveur.
* Mise en conformité du formulaire avec les exigences fonctionnelles du projet.
* Ajustement du design et de la responsivité en utilisant Foundation 5.5.1.

**Interactions :**

* Très peu d’interactions avec l’équipe, la majorité travaillant à distance aujourd’hui.
* Discussions légères avec un employé pendant la pause déjeuner.

**Problèmes rencontrés :**

* Plusieurs difficultés avec Mustache pour la validation des formulaires.
* Problèmes d’affichage et de structure du formulaire nécessitant des corrections multiples.
* Bugs liés à l'affichage du menu latéral et du bouton "hamburger" sur mobile.
* Difficulté à comprendre certains comportements de Foundation 5.5.1.
* Problème de surcharge CSS impactant l'affichage du menu.

**Conclusion :**

* Après plusieurs heures de corrections et d'ajustements, les formulaires sont désormais fonctionnels et validés.
* Le design est amélioré, mais quelques ajustements peuvent encore être nécessaires.
* Il reste à tester davantage les interactions utilisateur pour assurer une expérience fluide.

**24 mars 2025 – Réunion d'équipe et travail sur Foundation 5.5.1**

**Tâches réalisées :**

* Réunion avec l’équipe : présentation de l’avancement, discussion sur les objectifs de la semaine prochaine et clarification des tâches à venir.
* Explication de l’implémentation prévue pour la page d’annonces destinée à afficher toutes les annonces.
* Travail spécifique sur l’intégration du framework Foundation 5.5.1 pour la mise en œuvre du design frontend : création d’un header fixe, d’une sidebar positionnée sous le header (affichée sur desktop) et d’un menu hamburger pour mobile.
* Tests et ajustements CSS afin de résoudre les problèmes de positionnement et d’affichage sur desktop.

**Interactions :**

* Échanges constructifs lors de la réunion d’équipe, avec un focus sur la planification des développements à venir.

**Problèmes rencontrés :**

* Quelques difficultés avec le positionnement de la sidebar dues aux règles de position: fixed, résolues par l’ajustement du top (correspondant à la hauteur du header) et de la gestion des marges sur l’inner-wrap.

**Conclusion :**  
La partie design avec Foundation 5.5.1 est désormais fonctionnelle avec un header fixe et une sidebar correctement positionnée sous le header sur desktop. La prochaine étape consistera à implémenter la page d’annonces pour afficher toutes les annonces.

**25 mars 2025 – Amélioration du design, recherche géographique et gestion des filtres**

**Tâches réalisées :**

* Résolution d’un problème critique où les filtres de tri (par ville, par titre, par date) redirigeaient vers la mauvaise route.
* Finalisation de l’affichage des annonces avec image miniature, icône de fichier, et filtrage dynamique côté serveur selon les paramètres de l’URL (type, tri, recherche).
* Amélioration de l’interface utilisateur inspirée de Facebook Marketplace : intégration d’un champ de recherche de ville avec rayon, dans une approche similaire à celle du Marketplace de Facebook.
* Ajout d’un filtre ville + rayon simulé, avec possibilité future de tri géographique précis à l’aide de coordonnées.
* Ajout des colonnes latitude et longitude dans la base de données pour préparer l’intégration géographique.
* Intégration de Geoapify pour l’autocomplétion des villes et récupération automatique des coordonnées GPS.
* Résolution d’un bug où les champs latitude et longitude restaient vides si l’utilisateur tapait une ville sans cliquer sur la suggestion.
* Ajout d’une validation serveur pour empêcher la soumission d’annonces sans coordonnées géographiques valides.
* Débogage approfondi via error\_log() et inspection du payload $\_POST, correction de champs manquants.

**Interactions :**

* Discussion en fin de journée avec le team lead concernant l’approche géographique : décision que les annonces doivent rester limitées à la région de Lanaudière.
* Suite à cette discussion, remise en question de l’utilisation d’une API géographique payante : Geoapify ne couvre pas tous les noms de ville québécois (ex : Notre-Dame-des-Prairies), ce qui remet en cause son utilité.
* Proposition retenue : continuer d’offrir une expérience enrichie avec autocomplétion, mais permettre aussi une validation manuelle si la ville n’est pas détectée.

**Problèmes rencontrés :**

* Bug de formulaire dû à la duplication des champs latitude/longitude (présents deux fois dans le HTML).
* Aucune validation côté client si l'utilisateur entre une ville manuellement sans cliquer dans la liste des suggestions.
* Limitations majeures de la base de données Geoapify : certains noms de ville valides ne sont pas reconnus ou suggérés.
* Difficulté à faire cohabiter validation stricte et flexibilité pour l’utilisateur final.

**26 mars 2025 – Suppression de Geoapify, refonte complète du filtrage géographique**

**Tâches réalisées :**

* Refonte complète du système de filtrage par ville et rayon.
* Suppression de Geoapify suite à la confirmation que tous les utilisateurs sont basés dans Lanaudière (Québec).
* Création d’une nouvelle table localites contenant toutes les villes, municipalités, paroisses et villages de Lanaudière avec leurs latitudes, longitudes, MRC et préfixes postaux.
* Recherche manuelle exhaustive pendant 2 heures pour collecter les codes postaux et la classification municipale pour les 70 localités.
* Mise à jour du formulaire soumission\_offre pour utiliser la localiteId, et auto-remplir ville, mrc et codePostal via JavaScript.
* Modification du contrôleur annonceController::createAnnonce() pour récupérer les données de la localité et éviter toute dépendance API.
* Refonte de getAnnoncesByType() pour filtrer dynamiquement les annonces selon la ville choisie et un rayon en kilomètres (Haversine).
* Ajout d’un affichage dans l’interface utilisateur qui rappelle la ville choisie et le rayon appliqué.
* Refonte complète des pages liste\_offres.html et liste\_besoins.html avec un nouveau design responsive inspiré du Marketplace Facebook.
* Ajout d’un système pour afficher les cartes d’annonces par rangées de 4, avec ajustement CSS.
* Adaptation de la page soumission\_besoin.html pour intégrer les mêmes règles de localité que soumission\_offre.html.

**Interactions :**

* Aucune interaction avec l’équipe aujourd’hui.
* J’étais seul sur place et agissais comme référent technique de la journée.

**Problèmes rencontrés :**

* J’ai perdu beaucoup de temps à comprendre la structure municipale du Québec : il existe plusieurs types d’entités (ville, paroisse, municipalité, village) regroupées en MRC.
* Difficulté initiale à récupérer les préfixes postaux exacts pour chaque localité.
* Conflits de validation dans les formulaires dus à des noms de champs qui ne correspondaient plus après la refonte.
* Problème critique corrigé : codePostal était NULL car le champ caché n’était pas correctement peuplé avant envoi.
* J’ai dû tout tester manuellement après chaque modification pour m’assurer que la validation, le filtrage et les requêtes SQL fonctionnaient comme prévu.

**Conclusion :**  
La transition hors de Geoapify est maintenant terminée. Le système de géolocalisation est 100% local, performant et stable. Toutes les fonctionnalités de recherche, soumission et affichage d’annonces sont compatibles avec cette nouvelle architecture. Le projet est désormais aligné avec les exigences spécifiques du client (limitation à Lanaudière).

**27 mars 2025 – Upload d’images avec compression, galerie et miniature**

**Tâches réalisées :**

* Implémentation d’un **plugin JavaScript réutilisable** pour le téléversement d’images :
  + Sélection d’images avec aperçu instantané (jusqu’à 4 fichiers).
  + Redimensionnement automatique des images à un maximum de 1200x1200 pixels.
  + Compression automatique si la taille dépasse 20MB.
* **Intégration de CropperJS** pour permettre à l’utilisateur de sélectionner une image comme miniature :
  + Affichage d’un bouton “📸 Miniature” sous chaque image.
  + Affichage d’un modal Reveal (Foundation 5.5.1) contenant l’outil de recadrage.
  + Application d’un crop en 1:1 avec surlignage visuel de l’image sélectionnée comme miniature.
* **Ajout du champ d’upload** dans le formulaire de soumission d’une offre.
* **Sauvegarde des fichiers uploadés dans /public/uploads** avec :
  + Génération d’un nom unique.
  + Sauvegarde en base de données (media) du chemin, type MIME, URL publique, et flag is\_thumbnail.
  + Ajout dynamique de la clé url dans le tableau media pour compatibilité avec Mustache.
* **Affichage de la miniature** dans la page liste\_offres via la propriété thumbnail.
* **Affichage d’une galerie d’images** dans la page de détail d’une annonce avec prise en charge conditionnelle :
  + Image / Vidéo / PDF.
  + Fallback visuel si aucune image n’est présente.
* **Correction d’un bug bloquant** lié à l’appel à str\_starts\_with() (incompatible avec PHP < 8.0) remplacé par strpos($type, 'image/') === 0.

**Problèmes rencontrés :**

* Le **modal Reveal de Foundation 5.5.1** ne s’ouvrait pas correctement :
  + Parfois affiché au chargement, parfois impossible à ouvrir.
  + Le comportement bloquant (vraie popup) n’est pas encore totalement fonctionnel.
  + Tentatives avec data-reveal-id, aria-hidden, script JS et triggers foundation('open') sans succès complet.
* L’image recadrée avec CropperJS **ne remplaçait pas l’originale** pour le champ de miniature :
  + Nécessité de corriger la logique de substitution de fichier dans le champ <input type="file">.

**Interactions avec les membres de l’équipe :**

* Discussion avec le **responsable de stage** pour clarifier le fonctionnement attendu du **modal Reveal de Foundation 5.5.1**.
* Il m’a aidé à comprendre les particularités de l’implémentation modale avec Foundation (attributs requis, limitations connues, etc.), ce qui m’a permis d’avancer malgré les difficultés rencontrées.

**31 mars 2025 – Refonte complète du plugin d’upload d’images avec CropperJS**

**Tâches réalisées :**

* Refonte du système d’upload d’images en profondeur pour le rendre plus intuitif, interactif et réutilisable.
* Intégration complète de **CropperJS** pour permettre le **recadrage manuel** des images dans une modale responsive.
* Ajout de la sélection des images via **glisser-déposer** ou champ <input type="file">.
* Conception d’une interface divisée en deux zones distinctes : à gauche l’aperçu des images téléchargées, à droite la zone de recadrage.
* Implémentation de la **compression automatique** et du **redimensionnement** des images si nécessaire (max 4 fichiers).
* Création d’une **nouvelle interface utilisateur** plus intuitive avec bouton de confirmation du recadrage et rafraîchissement dynamique des vignettes.
* Intégration de cette fonctionnalité dans le formulaire de soumission d’annonces, avec gestion du **blob original** et du **blob recadré**.
* Utilisation de **Foundation 5.5.1** pour la gestion modale du recadrage.

**Interactions :**

* **Réunion d’équipe lundi matin** : partage de l’état d’avancement de la semaine précédente et objectifs de la semaine (compléter ce plugin).
* **Échange avec mon responsable de stage** pour obtenir des conseils spécifiques sur l’implémentation du **modal Reveal** et les limites de Foundation.
* Conversation informelle à la fin de la réunion sur **l’organisation d’une fête surprise** pour une collègue (Andrée) qui part à la retraite.

**Problèmes rencontrés :**

* Comportement incohérent du modal Reveal : parfois affiché au chargement, parfois ne s’ouvrait pas.
* Difficulté à remplacer correctement le fichier original par le fichier recadré dans le champ <input type="file">.
* Nécessité de manipulations complexes avec les attributs aria-hidden, data-reveal-id, et déclenchements JavaScript pour un comportement fluide.

**Conclusion :**

La refonte du plugin est maintenant terminée et offre une expérience utilisateur moderne et flexible. Le système est modulaire, réutilisable et conforme aux exigences du projet. Il reste quelques ajustements mineurs à faire sur la gestion modale, mais le cœur du plugin est pleinement fonctionnel.

**4 janvier 2025 – Optimisation et finalisation de l’interface du plugin ImageUploader**

**Tâches réalisées :**  
• Ajout d’une grille de 4 tuiles (2x2) pour l’affichage des images uploadées, avec des placeholders pour les emplacements vides.  
• Implémentation d’un bouton de suppression sur chaque tuile pour retirer une image.  
• Mise en place d’une zone de recadrage intégrant CropperJS, avec un placeholder affiché lorsque aucune image n’est sélectionnée.  
• Ajout d’un bouton “Save Changes” en bas à droite pour finaliser la sélection et le recadrage des images.  
• Application de styles modernes et minimalistes aux boutons (avec effets hover, couleurs flat et bords arrondis) ainsi qu’à la modale, tout en assurant la compatibilité avec Foundation 5.5.1 Reveal.  
• Révision et ajustement de la gestion des événements pour améliorer la fluidité de l’interface modale.

**Interactions :**  
• Travail en autonomie depuis chez moi.

**Problèmes rencontrés :**  
• Conflits entre les styles personnalisés et le comportement par défaut de Foundation Reveal, nécessitant des ajustements pour la réactivité du modal.  
• Ajustements nécessaires pour éviter des espaces indésirables entre les tuiles et garantir l’alignement parfait de l’interface.

**Conclusion :**  
L’optimisation de l’interface du plugin ImageUploader est désormais finalisée. L’expérience utilisateur est moderne, intuitive et réactive, et le plugin reste entièrement compatible avec Foundation 5.5.1 Reveal. Quelques réglages mineurs pourront être effectués par la suite en fonction des retours.

**4 Avril 2025 – Implémentation du plugin MediaUploader et échanges avec le team lead**

**Tâches réalisées :**

* Développement d’un nouveau plugin MediaUploader pour gérer l’upload de médias (vidéos, PDF, documents) avec une limite de 15 MB.
* Mise en place d’un système de découpage en chunks pour les fichiers volumineux et réassemblage côté serveur via un endpoint dédié (media\_endpoint.php).
* Création d’une interface modale inspirée du plugin d’image, avec affichage en tuiles utilisant Material Icons, barre de progression visuelle et bouton de suppression pour chaque média.
* Intégration de la fonctionnalité de suppression d’un média, avec envoi d’une requête AJAX pour supprimer les fichiers temporaires côté serveur.
* Ajustements et tests pour s’assurer que l’upload ne démarre qu’au clic sur le bouton « Télécharger » et que la barre de progression se met à jour correctement.

**Interactions :**

* Discussion avec le team lead afin de clarifier les exigences fonctionnelles du plugin MediaUploader, notamment sur la gestion du découpage, l’upload en chunks et le comportement attendu lors de la suppression d’un média.
* Échanges sur l’interface utilisateur pour que celle-ci soit intuitive et cohérente avec le plugin d’image existant.

**Problèmes rencontrés :**

* Difficultés initiales pour afficher correctement le bouton de suppression dans la modale, qui nécessitaient des ajustements CSS pour obtenir un bouton visible et esthétique.
* Gestion de la mise à jour de la barre de progression qui restait à 0 jusqu’à ce que l’upload soit déclenché par le clic sur « Télécharger ».
* Quelques ajustements dans la communication entre le front-end (JS) et l’endpoint PHP pour s’assurer que les réponses AJAX soient correctement traitées.

**Conclusion :**  
L’implémentation du plugin MediaUploader progresse bien et répond aux attentes du team lead. Le système est désormais capable de gérer l’upload de différents types de médias en découpant les fichiers volumineux et en affichant une barre de progression visuelle dans une modale réutilisable. Des ajustements mineurs restent à réaliser pour parfaire l’interface, mais le cœur du plugin est désormais fonctionnel et conforme aux exigences du projet.

**28 mars 2025 – Implémentation de la finalisation différée dans le plugin MediaUploader**

**Tâches réalisées :**  
• Séparation de la phase d’upload temporaire (stockage des chunks dans le dossier temp) de la finalisation définitive.  
• Retrait de l’appel immédiat à finalizeUpload() dans la fonction uploadFile(), afin que les chunks restent dans le dossier temp pendant la prévisualisation.  
• Ajout d’une fonction (ou d’un mécanisme) de finalisation globale qui sera déclenchée lors de la soumission finale du formulaire, afin de réassembler les chunks et de déplacer le fichier final dans le dossier uploads.  
• Ajustement des tests et des logs côté client pour vérifier que tous les chunks sont bien uploadés et que la progression se met à jour correctement.

**Interactions :**  
• Discussion avec le team lead pour clarifier que les médias ne doivent être définitivement enregistrés (transférés dans uploads) qu’après confirmation du formulaire, évitant ainsi la création de fichiers orphelins en cas d’abandon.

**Problèmes rencontrés :**  
• Compréhension et adaptation du workflow pour différer la finalisation de l’upload.  
• Ajustement du code JavaScript pour garantir que la finalisation ne se déclenche pas immédiatement après l’upload des chunks.  
• Mise au point du mécanisme de finalisation lors de la validation du formulaire, afin d’assurer que seuls les médias confirmés soient déplacés définitivement.

**Conclusion :**  
La finalisation différée dans le plugin MediaUploader est désormais implémentée. Les fichiers uploadés restent dans le dossier temporaire jusqu’à la soumission finale du formulaire, conformément aux exigences fonctionnelles définies avec le team lead. Des tests supplémentaires seront effectués lors de l’intégration avec le formulaire global pour s’assurer de la bonne cohérence du workflow.

**14 Avril 2025 – Mise en œuvre de la sauvegarde des liens médias en session et tests finaux du MediaUploader**

**Tâches réalisées :**  
• Intégration du stockage des URLs des médias finalisés dans une variable de session via le fichier endpoint (media\_endpoint.php) du plugin MediaUploader.  
• Modification du bloc de finalisation pour qu’après le réassemblage des chunks, le lien public (fileUrl) soit sauvegardé dans le tableau de session ($\_SESSION['uploadMediaFiles']).  
• Ajustement du code PHP pour démarrer la session (session\_start()) et s’assurer que le bloc de sauvegarde des liens soit exécuté avant l’appel à exit.  
• Réorganisation du workflow pour que les chunks soient uploadés et stockés temporairement dans le dossier « temp » et ne soient réassemblés dans le dossier « uploads » qu’après confirmation finale via la soumission du formulaire.  
• Ajout de logs supplémentaires côté serveur et dans le JS pour vérifier le bon déroulement de l'upload et de la finalisation, en particulier pour les fichiers volumineux (vidéos).

**Interactions :**  
• Réunion approfondie avec le team lead dans laquelle il a confirmé l’approche consistant à stocker les liens finaux dans la session afin de faciliter la réutilisation du plugin dans différentes parties du site.  
• Échange constructif avec l’équipe sur la présentation finale d’un autre stagiaire, permettant de comparer les techniques d’upload et de gestion des médias, et de partager des idées pour améliorer l’expérience utilisateur.

**Problèmes rencontrés :**  
• Difficulté initiale pour s’assurer que le bloc de finalisation exécutait la sauvegarde dans la session avant d’arrêter le script (problème de placement de session\_start() et d’ordre d’exécution).  
• Nécessité d’ajuster la logique de finalisation afin que les fichiers restants dans le dossier temporaire ne soient pas traités tant qu’aucune soumission finale n’est effectuée, afin d’éviter la création de fichiers orphelins en cas d’abandon du formulaire.

**Conclusion :**  
La fonctionnalité de sauvegarde des liens médias dans la session est désormais opérationnelle. Les tests montrent que, pour les fichiers volumineux, les chunks sont correctement uploadés et stockés temporairement, et que la finalisation (réassemblée et enregistrement définitif) s’effectue uniquement lorsque cela est déclenché (via la soumission finale du formulaire). L’orientation adoptée a été validée lors de la réunion avec le team lead, garantissant ainsi une intégration réutilisable du plugin MediaUploader dans diverses parties du site.

**15 Avril 2025 – Correction du comportement de finalisation pour les fichiers de petite taille et ajout de la gestion des extensions dans MediaUploader**

**Tâches réalisées :**  
• Modification du script JavaScript pour que, même pour les fichiers inférieurs à 15 MB (créant ainsi un seul chunk nommé "chunk\_0"), le processus de finalisation soit déclenché automatiquement via finalizeUpload.  
• Ajout de la fonction helper **getFinalFileName()** dans le JS afin de vérifier et compléter le nom du fichier avec une extension appropriée (en se basant sur le type MIME) lorsque le nom original ne la contient pas (exemple : "Untitled video – Made with Clipchamp").  
• Adaptation du fichier PHP de finalisation pour extraire l'extension du fichier à partir du nom fourni et l'enregistrer dans la session aux côtés de l'URL du fichier finalisé.  
• Correction du problème d'appel multiple de **session\_start()** en ajoutant une vérification de l'état de la session (via session\_status()).  
• Réalisation de tests sur différents types de fichiers (petits et volumineux) afin de valider que tous les fichiers, quelle que soit leur taille, sont correctement réassemblés et que leurs informations (URL et extension) sont bien stockées dans la variable de session.

**Interactions :**  
• Validation des modifications avec le team lead, qui a confirmé que le comportement attendu pour les fichiers de petite taille (finalisation automatique et gestion correcte de l'extension) correspondait aux besoins du projet.  
• Échanges avec l'équipe technique sur la gestion par défaut des extensions pour les fichiers dont le nom ne contient pas d'extension, aboutissant à la mise en place de la fonction helper dans le script JS.

**Problèmes rencontrés :**  
• Initialement, les fichiers de taille inférieure à 15 MB restaient dans le dossier temporaire car le processus de finalisation n'était pas déclenché pour un unique chunk.  
• Absence d'extension extraite pour les fichiers dont le nom ne comportait pas de point (exemple : "Untitled video – Made with Clipchamp"), nécessitant l'ajout de la fonction getFinalFileName() pour appliquer une extension par défaut (par exemple ".mp4").  
• Message d'erreur lié à des appels multiples de session\_start(), résolu en conditionnant l'appel à session\_start().

**Conclusion :**  
Le workflow du MediaUploader a été corrigé pour que tous les fichiers, qu'ils soient volumineux ou inférieurs à 15 MB, soient automatiquement finalisés et réassemblés dans le dossier "uploads". De plus, les informations essentielles (URL finale et extension du fichier) sont désormais correctement stockées dans la session, facilitant leur traitement ultérieur sur le serveur. Ces ajustements, validés par le team lead, garantissent une meilleure robustesse et une réutilisation optimale du plugin dans différentes parties du site.

**22 avril 2025 – Séparation par type de média et amélioration responsive du MediaUploader**  
**Tâches réalisées :**  
• Implémentation de sections distinctes pour chaque type de média dans la modale du plugin MediaUploader (📹 Vidéos, 📄 Documents).  
• Ajout d’un système de regroupement visuel avec titres et sections repliables (accordéon), permettant une navigation claire parmi les fichiers ajoutés.  
• Optimisation de l’interface responsive pour un affichage adapté sur tous les formats (mobile, tablette, desktop).  
• Ajout de règles CSS spécifiques pour assurer la responsivité et éviter les conflits avec Foundation 5.5.1.  
• Intégration correcte du plugin dans le cycle Reveal de Foundation, avec initialisation de la modale via $(modal).foundation() afin d’éviter les comportements imprévisibles au redimensionnement.  
• Mise à jour de la logique d’insertion des tuiles pour les répartir automatiquement dans la bonne section en fonction du type MIME.  
• Ajout d’une indication dynamique du nombre de médias par catégorie, mise à jour à chaque ajout ou suppression de tuile.

**Interactions :**  
• Participation à la réunion d’équipe du jour pour présenter l’état d’avancement des fonctionnalités MediaUploader.

**Problèmes rencontrés :**  
• Le comportement responsive initial du modal était instable à cause des règles CSS de Foundation.  
• Les tuiles perdaient leur alignement ou dépassaient dans les sections sans align-self: flex-start.  
• Il a fallu combiner du CSS personnalisé avec des sélecteurs spécifiques pour éviter que Foundation n’écrase les styles en mobile.  
• Réajustement du DOM pour éviter que le media-tile-container s’écrase lors du repli des sections avec .toggle-section.

**Conclusion :**  
L’interface du MediaUploader est maintenant structurée par type de média, intuitive et pleinement responsive. Les sections vidéos et documents sont claires, avec possibilité de repli et d’affichage dynamique. Le système est robuste, prêt pour intégration finale dans les formulaires, et conforme aux pratiques UX/UI modernes.

**28 avril 2025 – Vérification de l’intégration des médias et requêtes SQL**

**Tâches réalisées :**

* Test complet du plugin MediaUploader dans le formulaire de soumission d’annonces (vidéos, PDF, documents).
* Ajout et vérification de la récupération du champ *fileType* côté serveur (utilisation de mime\_content\_type dans media\_endpoint.php).
* Validation que les médias (image, vidéo, PDF) sont bien dispatchés via l’événement mediaUploader:uploadComplete et stockés dans la base.
* Création et test de la requête SQL pour afficher tous les médias liés à une annonce (ex. : SELECT … FROM annonces LEFT JOIN media ON media.annoncesId = annonces.id WHERE annonces.id = 60;).
* Vérification en phpMyAdmin que les enregistrements media sont corrects (chemin, URL publique, type MIME, flag is\_thumbnail).

**Interactions :**

* Réunion d’équipe matinale dédiée au point d’avancement du plugin MediaUploader (aucun autre échange particulier).

**Problèmes rencontrés :**

* Aucun obstacle majeur : le workflow d’upload, de finalisation et de liaison en base fonctionne désormais comme attendu.

**Conclusion :**  
Le plugin MediaUploader est désormais pleinement intégré au backend : chaque média est enregistré avec son type MIME, lié à l’annonce correspondante et récupérable via une requête SQL dédiée. Le flux de travail a été validé en équipe lors de la réunion du jour.