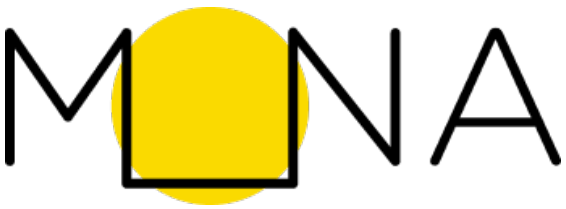
Rapport Final IFT3150

Miliya Ai
Département d'informatique et recherche opérationnelle
Université de Montréal

24 avril 2023

Table des matières



1 Introduction

Je participe à un projet dans le cadre du cours IFT3150, appelé MONA. Il s'agit d'une application mobile développée par la Maison MONA, une organisation à but non lucratif résultant d'une collaboration entre le Département d'histoire de l'art et d'études cinématographiques et le Département d'informatique et de recherche opérationnelle (DIRO) de l'Université de Montréal.

Conçue pour découvrir, photographier et collectionner l'art public, l'application mobile Mona donne accès à des bases de données contenant de l'information publique sur les œuvres artistiques, les sites culturels et le patrimoine du Québec, provenant d'autres bases de données publiques et encourage les utilisateurs à collectionner ces trois types de « découverte » en prenant des images et en les commentant.

Du côté de la recherche, il envoie des enregistrements de données de commentaires publics sur l'art (photos et commentaires) à un serveur, et nous y avons une interface utilisateur utilisée par les administrateurs pour visualiser ces données afin d'aider les chercheurs à comprendre comment le public évalue ces résultats.

2 Changements effectués pendant mon mandat

Je suis extrêmement honoré et enthousiaste d'avoir rejoint l'équipe de MONA en tant que développeur serveur. Mon rôle principal consiste à analyser l'architecture du système existant, à évaluer et améliorer le cadre existant, ainsi qu'à concevoir de nouvelles structures de base de données. Ces tâches représentent un défi significatif pour moi, car contrairement aux devoirs scolaires, il s'agit de ma première expérience avec un projet déjà bien établi, disposant de données réelles, d'utilisateurs actifs et de besoins concrets. Il est nécessaire que je comprenne rapidement le système en place et que je cherche les domaines nécessitant des améliorations.

Bien que j'aie participé à un projet de stage durant l'été 2023, où nous avions démarré de zéro, travaillé chez MONA m'a obligé à ajuster mes stratégies et à me familiariser avec de nouveaux outils, notamment le langage de programmation PHP et le framework Laravel. Cette transition nécessite une adaptation rapide et une compréhension approfondie des enjeux spécifiques à un environnement de production en direct, ce qui ajoute à la complexité, mais également à l'enrichissement de mon expérience professionnelle.

Un grand merci à mon équipe ! La réunion de chaque semaine offre une nouvelle opportunité d'apprendre et de m'adapter, renforçant ainsi mes compétences en développement et ma capacité à contribuer efficacement à l'évolution de l'application MONA. Ce projet, par son ampleur et son impact potentiel, représente une étape

cruciale de ma carrière, me permettant de mettre en pratique mes connaissances théoriques dans un contexte dynamique et exigeant.

2.1 Analyse

Au début du projet, ma tâche principale était d'analyser l'infrastructure du projet (créer un diagramme de structure de serveur sur Miro), de découvrir les parties peu conviviales de l'interface utilisateur d'administration et d'annoter le code correspondant. De plus, j'ai participé aux tests de la version mobile d'Ionic, ajusté l'emplacement de stockage des rapports unifiés, et participé à la préparation de tutoriels pour optimiser le processus d'installation pour les futurs développeurs (l'utilisation du protocole SSH, et la mise en place de l'environnement de développement local).

1. Test de la version mobile d'Ionic

J'ai participé aux tests de la version mobile d'Ionic pour m'assurer que les fonctionnalités et l'expérience utilisateur répondaient aux exigences du projet. Les problèmes rencontrés lors du test de reconnaissance incluent l'impossibilité de télécharger des photos sur l'album local du téléphone mobile, l'impossibilité de revenir à l'étape précédente après avoir cliqué sur le bouton de l'appareil photo et l'impossibilité de télécharger des photos sans cliquer sur le bouton d'envoi. Ma collègue Kim sera responsable du travail de suivi sur la vulnérabilité

2. Ajustement unifié de l'emplacement de stockage des rapports

J'ai ajusté l'emplacement de stockage des rapports de projet unifiés pour améliorer l'accessibilité et l'organisation. Cependant, le serveur fonctionne actuellement anormalement et ne peut pas télécharger de fichiers directement, je l'envoie donc à Simon à chaque fois et il le met à jour manuellement.

3. Création du didacticiel du processus d'installation du développeur

J'aide à créer des tutoriels conçus pour optimiser le processus d'installation pour les futurs développeurs, notamment en développant des tutoriels couvrant l'utilisation du protocole SSH et des instructions pour la mise en place d'un environnement de développement local. Ces didacticiels sont conçus pour simplifier le processus d'intégration des nouveaux développeurs et sont présentés sur un wiki github pour fournir une référence documentée pour le flux de travail de développement.

4. Commentaires sur le code de l'interface utilisateur :

L'identification des aspects peu conviviaux de l'interface utilisateur d'administration est cruciale pour améliorer l'expérience utilisateur globale. J'ai donc commenté une partie du code qui n'était pas peaufiné et ajouté une explication de la fonctionnalité à compléter à côté. Les équipes de développement ultérieures pourront prioriser les améliorations de l'interface utilisateur et mettre en œuvre des principes de conception centrés sur l'utilisateur.

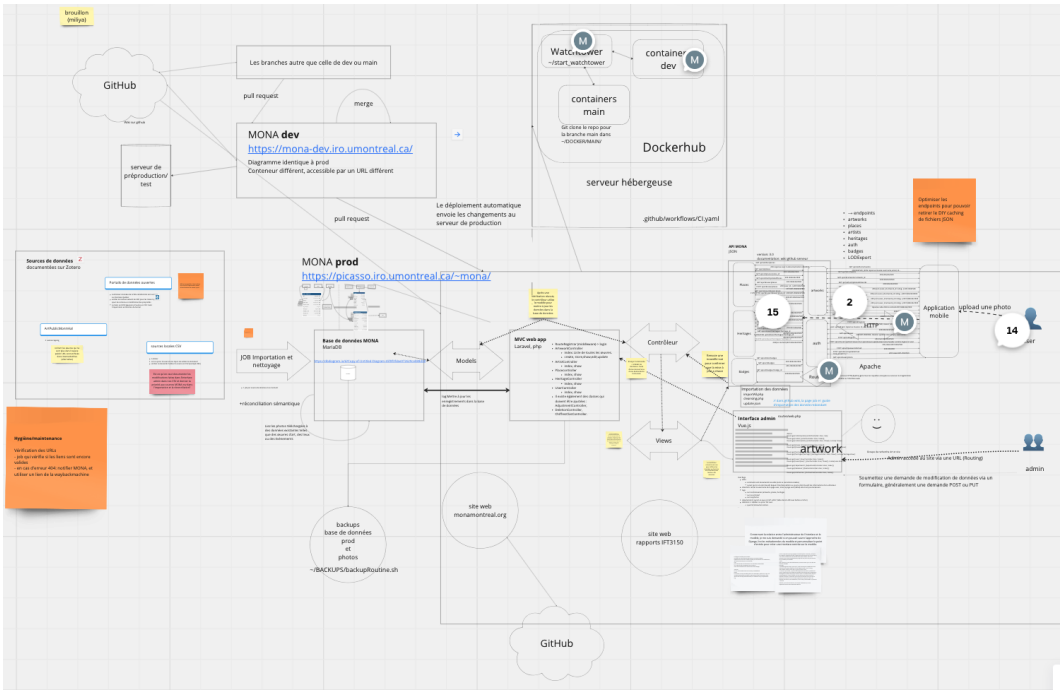


FIGURE 1 – Capture d'écran de l'infrastructure de Miro

5. L'infrastructure

La structure du serveur du projet a été analysée et documentée en profondeur à l'aide de Miro. L'objectif est de déterminer l'état actuel de notre infrastructure, d'examiner les flux de déploiement et de garantir que les serveurs répondent aux exigences d'évolutivité et de fiabilité du projet. Le diagramme Miro créé sert de représentation visuelle et de référence à l'équipe pour comprendre les interconnexions entre les différents composants tels que GitHub, Dockerhub, les environnements de développement (dev) et de production (prod), ainsi que les pipelines d'intégration et de livraison continues. Configuration du serveur : Notre configuration de serveur se compose d'environnements de développement et de production distincts, chacun accessible via une URL différente. Cette isolation fournit un environnement de production stable tout en fournissant un bac à sable pour le développement et les tests. Déploiement automatique : mise en œuvre du processus de déploiement automatique et application des modifications au serveur de production, améliorant ainsi l'efficacité de notre déploiement.

En analysant l'infrastructure, nous avons découvert de nouvelles exigences, par exemple, si un utilisateur change de téléphone et demande à l'API de repeupler son application, il doit obtenir les données les plus récentes (dernières modifications). Si l'administrateur va sur la page d'un utilisateur, nous voulons voir l'historique

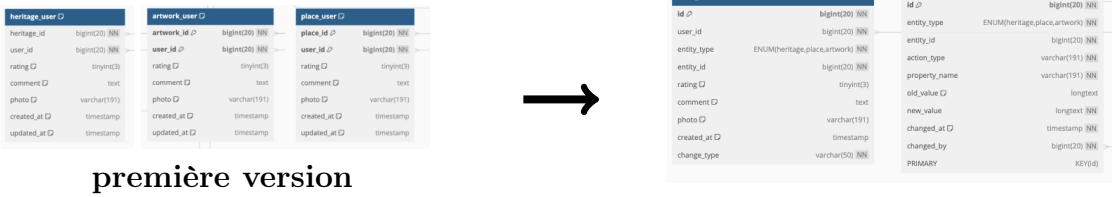
des modifications (daté). Si un administrateur accède à une page d'une œuvre où il y a plusieurs photos du même utilisateur, nous voulons voir toutes les photos. Si les données sont réimportées, les modifications de l'administrateur ne doivent pas l'être écrasé par la version de la base de données. Les systèmes de contrôle sont conçus pour suivre et enregistrer l'historique des modifications des données. De plus, si les sources de données ouvertes (Portails de données ouvertes, ArtPublic-Montréal) sont supprimées ou modifiées, cela aura également un impact sur notre base de données et sur la recherche. Ce fut une perte énorme, alors mon collègue a pris en charge la tâche de transférer les données ouvertes vers un CSV local.

Le processus d'analyse mentionné ci-dessus nous a conduit à décider de développer un système de contrôle de version centralisé adapté à toutes les options susmentionnées (optimisation du stockage des données dans la base de données), ce qui constituera précisément mon travail à venir → la conception.

2.2 Conception

Pour la deuxième phase du travail, j'ai principalement conçu et modélisé un système de contrôle de version centralisé.

La figure ci-dessous montre quelques modifications apportées à la base de données :



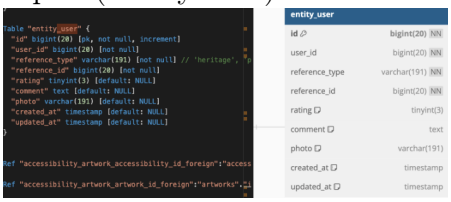
première version **version améliorée**

Il est possible de fusionner 'HeritageUser', 'placeUuser', 'artworkUuser' en une seule grande table, et la fusion peut avoir plus de sens dans ce cas. Les interactions des utilisateurs avec 'artwork', 'places' et 'heritages' (par exemple, rating, comment, etc.) peuvent être représentées en utilisant la même structure de données.

Et afin de garantir que l'historique du système est enregistré dans la base de données, j'ai conçu UserContent ,EntityHistories pour stocker respectivement les modifications du client et des données ouvertes.Expliquez plus attentivement, champ EntityId : le type de données du champ EntityId dans la table EntityHistories doit correspondre au type de données du champ id de la table d'entité à laquelle il fait référence (œuvres d'art, lieux, patrimoines). Étant donné que l'entityId fait référence à plusieurs tables différentes, il ne peut pas être directement défini comme clé étrangère au sens traditionnel (car les clés étrangères pointent

généralement vers une table spécifique). Par conséquent, la logique de l'application doit être complétée pour garantir l'exactitude de l'entitéType et de l'entitéId.

C'est un nouveau problème qui est apparu, qu'il s'agisse de créer une table d'historique spéciale (‘entityUserHistories‘)pour suivre les modifications des données ou de modifier directement la table d'origine pour inclure des données historiques (‘entityUser‘) ?



EntityUser

?

EntityUserHistories

Après avoir examiné les informations, j'ai découvert que si notre système est davantage axé sur l'intégrité et la sécurité de l'historique des données et qu'il est censé interroger fréquemment les données actuelles et interroger les données historiques moins fréquemment, la création d'une table d'historique dédiée peut être un meilleur choix. . Au contraire, si la conception doit être simplifiée et a moins d'impact sur les performances des requêtes, ou si la fréquence des modifications des données n'est pas élevée, il peut alors être plus approprié de modifier la table d'origine. Ensuit, pour limiter le type d'entité à stocker uniquement trois valeurs de chaîne spécifiques ("patrimoine", "lieu", "illustration"), j'ai changé le type de colonne de varchar en un type ENUM. Le type ENUM permet de définir l'ensemble de valeurs qu'une colonne peut accepter. Le champ changeHistory est un champ de type JSON qui peut être utilisé pour stocker un tableau d'enregistrements d'historique des modifications. Chaque enregistrement peut inclure le type de changement, la date à laquelle le changement s'est produit et éventuellement d'autres informations pertinentes. L'utilisation des types JSON vous offre la flexibilité d'enregistrer une variété de détails sans ajouter de champs distincts pour chaque élément d'information possible.

De plus, j'ai constaté que la méthode index() récupère tous les enregistrements « supprimés en douceur » des modèles « Artwork », « Place » et « Heritage ». « Suppression logicielle » signifie que l'enregistrement est marqué comme supprimé mais n'est pas réellement supprimé de la base de données, permettant ainsi la récupération. Il utilise la méthode "onlyTrashed" de l'ORM Eloquent de Laravel pour obtenir ces enregistrements. Il charge plusieurs relations par modèle, ce qui indique un modèle de données complexe avec de nombreuses entités interconnectées. Après avoir obtenu les enregistrements, il les convertit en JSON à l'aide des classes « ArtworkResource », « PlaceResource » et « HeritageResource », qui peuvent être des classes de ressources personnalisées utilisées pour la conversion d'API. Enfin, il renvoie une vue nommée « admin.deletions.index », transmettant la ressource

	entity_user_histories	entity_user
Isolation des données	séparez les données actuelles des données historiques pour garder les tables métier propres et efficaces. Cela permet d'optimiser les performances des requêtes, en particulier les requêtes sur les données actuelles.	à mesure que les données historiques augmentent, la table deviendra de plus en plus grande
Flexibilité	librement concevoir la structure de la table d'historique selon vos besoins, par exemple, enregistrer uniquement les modifications dans les champs clés ou ajouter des informations d'audit supplémentaires (telles que la raison du changement, le type de changement, etc.)	l'interrogation de données actuelles ou de données à un moment précis peut nécessiter une logique de requête plus complexe, en particulier lorsque les données historiques sont mélangées aux données actuelles.
Consommation d'espace	les données historiques peuvent occuper une grande quantité d'espace de stockage.	l'état actuel et l'état historique des données sont stockés dans la même table, ce qui facilite l'interrogation de l'historique complet des données.
Complexité de la maintenance	une logique supplémentaire est requise pour gérer l'historique des données	aucune table supplémentaire ni logique complexe n'est requise pour suivre les modifications des données

FIGURE 2 – Avantages et inconvénients

codée JSON à afficher, éventuellement sous forme d'application JavaScript ou de données rendues côté serveur. Les autres méthodes (‘create‘, ‘store‘, ‘show‘, ‘edit‘, ‘update‘, ‘destroy‘) sont des stubs généralement utilisés pour les contrôleurs de ressources dans Laravel, mais dans ce cas, ils ne sont pas implémentés (vides). L'interface utilisateur correspondante peut afficher une liste d'œuvres d'art, de lieux et d'héritages supprimés et fournir des options pour les supprimer ou les restaurer définitivement, car le contrôleur gère les éléments supprimés de manière logicielle. De plus, selon AdjustmentController, j'ai trouvé qu'il ne s'agissait que d'un modèle de contrôleur dans Laravel. Je suppose donc que les développeurs backend de l'époque voulaient laisser un modèle vide pour développer d'autres modèles. Par exemple, index() convertit toutes les données de modèle « adaptées » dans la base de données en JSON, puis les transmet au view.index de l'administrateur. ajustements pour l'affichage des données dans la liste, mais évidemment nous ne "réglons" pas le modèle en BD.

Nous avons décidé de ne pas utiliser la table userContentHistories pour suivre les modifications historiques et souhaitions mettre à jour les données directement dans la table userContent tout en conservant un enregistrement des modifications. En proposant et en optimisant continuellement plusieurs plans d'amélioration du système, nous avons discuté de la méthode d'utilisation de la fonction de suppression logicielle dans le cadre Laravel pour optimiser le mécanisme de traitement et de récupération des données. La structure finale des entitésHistories et userContent introduites dans la base de données est la suivante.

entity Histories Final

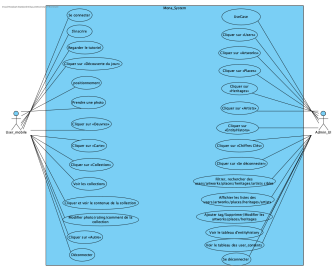
user Content Final

Appliquer ces changements sur UI :

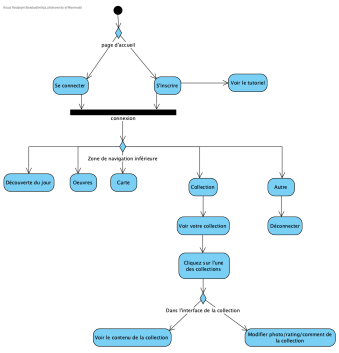
Après une évaluation approfondie de la structure de la base de données existante, j'ai procédé à une série de réajustements de l'interface utilisateur. L'un des changements majeurs est l'introduction de la table EntityHistory, qui se situe au même niveau que les tables User, Artworks et autres dans la barre de navigation. Cette nouvelle structure permet un accès facile à l'historique des modifications à l'aide de filtres par EntityId et par type, ce qui facilite la visualisation des modifications historiques.En ce qui concerne les utilisateurs, nous n'avons pas le droit de modifier ou de supprimer les informations relatives aux utilisateurs, j'ai donc supprimé les boutons correspondants. UserContent est construit comme une table autonome, améliorant ainsi la traçabilité et la gestion des données historiques.

Développement de diagrammes UML :

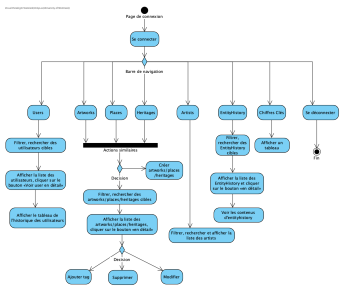
Pour m'assurer que les futurs développeurs comprennent et maintiennent mieux le système, j'ai créé des diagrammes d'activités et des diagrammes de cas d'utilisation en UML. Ces outils de visualisation facilitent la visualisation des interactions entre les utilisateurs et les systèmes et clarifient les processus au sein d'une application.



diagrammes de cas d'utilisation



diagrammes d'activités d'utilisateur



Admin d'interface

3 Vision Futures

J'ai vraiment hâte de voir la conception de mon interface utilisateur et la nouvelle structure de base de données mises en œuvre et déployées. De plus, le travail doit encore se poursuivre pour nettoyer la structure de la base de données et supprimer les tables en double afin d'optimiser les performances et l'efficacité du système.

4 Conclusion

Rejoindre l'équipe MONA est à la fois un défi et une belle opportunité d'apprentissage. Malgré des difficultés initiales de compréhension du projet et quelques barrières de communication, l'expérience m'a permis de persévérer et de renforcer mes compétences. Ce projet m'a permis d'apprendre le nouveau langage PHP et le framework Laravel, et m'a en même temps donné l'opportunité d'entrer en contact avec de vrais utilisateurs et données, d'analyser, de repenser et d'optimiser. Je suis très reconnaissant à l'équipe MONA pour l'opportunité et l'aide qui m'ont permis de vivre cette expérience enrichissante.