Rapport IFT 3150 - Projet d'informatique

MONA-Serveur

Vincent G. Beauregard

20034236

vincent.g.beauregard@umontreal.ca

Présenté au professeur
Guy Lapalme

DIRO
Université de Montréal
Canada
Hiver 2018

1 Introduction

Ce projet, initié dans le cadre du cours IFT3105 Interface personne machine par Daniel Jimenez et Lena Krause en 2015, représente aujourd'hui un projet d'envergure pour le département de l'histoire de l'art impliquant des étudiants de différents départements, notamment d'informatique. Le projet consiste à créer une application mobile qui sera distribuée via l'App Store (iOS) et Google Play(Android) où les utilisateurs pourront photographier et critiquer les oeuvres d'art public. L'application aura pour principale fonctionnalité d'identifier les différentes oeuvres à travers Montréal et de permettre aux utilisateurs de partager sur les réseaux sociaux leur découverte. Ainsi, il sera possible d'accumuler des données concernant la rétroaction du public par rapport à l'art, ce qui est peu documenté de nos jours.

En Automne 2017, Lena Krause a approché des étudiants en informatique dans le cadre du cours IFT3150 Projet d'informatique, afin de lancer le projet. Une petite équipe de programmeurs s'y est joint, soit Vincent G. Beauregard qui s'occupera du serveur, Émile Labbé qui s'occupera de la programmation Android et Paul-Philippe Chaffanat qui s'occupera de la programmation iOS, tous sous la direction de Guy Lapalme, professeur au DIRO. Au cours de la session d'hiver 2018, les programmeurs et Lena Krause se sont rencontrés hebdomadairement afin de discuter de l'avancement du projet et afin de fixer les différents objectifs pour les semaines à venir. Il est possible de retrouver le développement du projet sur la page web du projet où les étudiants entraient régulièrement de cours rapport.

2 Objectif

D'abord, une page web contenant l'ensemble des rapports des programmeurs devra être initialisée. Chaque programmeur pourra alors envoyer son rapport régulièrement au gestionnaire du serveur pour qu'il puisse garder la page à jour. Une section *Information générale* pourra contenir les renseignement sur nos réunions hebdomadaires à laquelle tous les programmeurs peuvent contribuer.

L'application mobile aura pour nécessité de se synchroniser à un serveur afin d'assurer le bon fonctionnement de l'application et surtout pour permettre aux professeurs du département de l'histoire de l'art de récupérer les critiques des utilisateurs concernant les oeuvres qu'ils auraient photographiées.

Via le serveur, il devra être possible de récupérer toutes les données nécessaires afin de faire rouler l'application, à l'exception de l'application elle-même. Ainsi les `API` disponibles sur le portail de données ouvertes de la ville de Montréal concernant l'art publique dans la ville de Montréal devront être rassemblés et normalisés. Il a été proposé de rassembler les données du bureau d'art public de la ville de Montréal via l' `API` `JSON` d'information sur les oeuvres de la collection municipale et les données du Programme d'art murale via l' `API` `GEOJSON` des

Murales subventionnées par la ville de Montréal.

Il devra également être possible de récupérer les données de chaque utilisateur au cas ou un utilisateur perdrait ses informations ou changerait simplement d'appareil mobile. Une base de données devra rassembler l'ensemble des informations concernant l'interaction d'un utilisateur avec une oeuvre.

Finalement, il devra être possible de récupérer les données de chaque utilisateur de façon éthique pour les professeurs en histoire de l'art. Une page web sécurisé par un mot de passe unique devra se synchroniser à la base de données.

3 Ressources utilisées

Les serveurs du DIRO, plus précisément sur `http://www-etud.iro.umontreal.ca`, combleront les besoins de synchronisation à l'aide des service `PHP` et `MySql` qui y sont disponibles. Comme le language `PHP` à été présenté dans le cadre du cours IFT 3225 - Technologie de l'Internet et comme l'interaction avec les SGBD comme `SQL` à été présenté dans le cadre du cours IFT 2935 - Bases de données, ces ressources du DIRO en font un choix optimal. La récente mise à jour des serveurs permet l'utilisation de la version 7.2.0 de `PHP` et la version 5.5.52-`MariaDB` pour `MySql`.

Afin de tester en local les différentes fonctionnalités, les services proposés par XAMPP seront exploités. Ils permettent de rassembler facilement les différentes fonctionnalités de `Apache`, `PHP`, `MariaDB` et `Perl`. À plus long terme, il faudra déplacer les données vers un autre serveur privé afin de prévoir l'expiration du compte étudiant.

4 Implémentation

Initialement, de simples tests ont été implémentés afin d'assurer que la communication s'effectuait entre les plate-formes, ce qui fut un succès. Le fichier test_get testait la connexion à l'équivalent d'un `Hello World!`. Il communiquait directement avec la base de données. Ce qu'il faisait : réception d'un `username` , vérification de son existence dans `MySql`, puis envoie du résultat sous la forme `Name:` *`username`* `- password:` *`password`*. Il était possible de tester la version `GET` en visitant la page web `http://www-etud.iro.umontreal.ca/~beaurevg/ift3150/test/test_get.php?username=Paul` qui prenait, pour le paramètre `username` , la valeur `Paul` qui fait partie de la base de données de test.

La seul problématique considérable à ce niveau fut le changement de la version de `PHP` sur les serveurs, peu avant la session d'hiver 2018. La nouvelle version de `PHP` contenait quelques problèmes de rétro-compatibilité. Les fonctions disponibles, plus particulièrement celle communiquant avec `Myql`, ont dû

être révisées afin de pouvoir appliquer les tests initiaux.

Avec la confirmation de la réception avec succès des plate-formes mobiles, il fut possible de débuter le développement au niveau du serveur.

4.1 Rassemblement des `API`

Cette section fut la première étape du projet à être réalisée. Elle consistait à rassembler en un fichier les données des `API` d'art public et de murales en un unique fichier `JSON`. Le fichier est généré en juxtaposant des chaînes de caractères. Il contient une liste d'oeuvres ayant les attributs suivants: `id`, `Titre`, `TitreVar`, `Categorie`, `CategorieANG`, `SousCategorie`, `SousCategorieANG`, `Date`, `Materiaux`, `Technique`, `Dimension`, `Arrondissement`, `Latitude`, `Longitude` et `Artiste`. `Materiaux` et `Technique` contiennent tous les deux une table où chaque élément contient les attributs `Nom` et `NomANG`. Pour ce qui est de l'attribut `Artiste`, celui-ci contient une table où chaque élément contient les attributs `Prénom`, `Nom` et `NomCollectif`.

Une grande partie du travail consistait à changer la représentation d'un attribut des murales pour qu'il concorde avec le format proposé (format qui se rapprochait beaucoup du fichier d'art public). Les artistes, par exemple, nécessitaient un tri vigoureux afin de gérer les multiples variations possibles. La séparation de multiples artistes, la présentation d'un nom de regroupement d'artiste et même le nom de certain artiste variait selon l'oeuvre. L'ajout d'un identifiant pour chacun d'entre eux à légèrement complexifié le tout. L'analyse n'est pas encore parfaite en tout point, mais on note une nette amélioration de la normalisation après ce filtre.

L'ajout des arrondissements aux oeuvres provenant du fichier des murales à nécessité l'interaction avec l' `API` des limites administratives de l'agglomération de Montréal offrant les polygones de chaque arrondissement. Un algorithme recherche si les coordonnées d'une murale se trouvent dans le polygone d'un arrondissement. Grossièrement, l'algorithme exploite la propriété que si un segment de droite partant d'un point et allant à l'infini croise les côtés du polygone un nombre impair de fois, ce point est à l'intérieur du polygone.

La normalisation des attributs `Technique` à été reportée dû à l'incohérence quasi-complète entre chacun d'entre eux.

Malgré le bon fonctionnement des précédentes tâches cités, certaines complications ont ralenti le travail au cours de la session, nécessitant des discussions avec l'équipe de programmation.

Une première complication fut révélée lors d'un premier test échoué qui consistait à créer un fichier `.txt` sur le serveur. La solution à été bêtement résolu plus tard en corrigeant les permissions des fichiers parents, mais en début de

parcours, la solution à mal été recherchée et mal résolue, soit en retournant le fichier après analyse à chaque requête plutôt que d'envoyer un fichier simplement contenu sur le serveur.

Une seconde complication fut lorsque nous avons constaté que l' `API` des murales ne contenait aucun titre. Après avoir consulté Lena Krause à ce sujet, il a été conclu que la majorité des murales n'aurait pas de titre afin de laisser au public la chance de nommer eux-même l'oeuvre. La section `Titre` des murales contiendra alors `NULL` en tout temps dans notre `API`.

Une troisième complication se présenta lors de l'analyse des dates. Le choix entre convertir les date en format `Unix TimeStamp` comme dans le fichier d'art public ou en format de date plus conventionnel (`yyyy-mm-dd`) fut délimité lors d'une rencontre où les programmeurs mobiles ont déclaré avoir une préférence pour le format `Unix TimeStamp` afin d'optimiser le tri par date de leur côté.

Une dernière complication fut au niveau de la validité du fichier. Bien que le convertisseur de fichier `JSON` laissait passer les premières versions du fichier, le fichier était illisible pour les programmeurs mobiles. Il a donc été nécessaire de passer le fichier dans un validateur de fichier `JSON` afin de détecter les erreurs qui furent corrigées en passant l'ensemble des entrées dans un filtre de symbole menant à la corruption du fichier.

Finalement, le fichier est disponible à tous via la page `http://www-etud.iro.umontreal.ca/~beaurevg/ift3150/server/?request=loadJson1`. Il contient encore quelques impuretés qui devront être corrigées tels que des noms d'artistes qui varient que par un accent, ou des artistes qui semblent inventés de toute pièce. Malgré leur apparence, ils sont bien valides, mais peut-être serait-il préférable d'utiliser des noms complets plutôt que des surnoms tels que *Lorem Ipsum*, *123Klan* ou *Osti One* qui pourraient choquer les utilisateurs de l'application. Encore faut-il respecter les droits des artistes afin d'éviter toutes complications judiciaires.

4.2 Création de Base de données

Cette seconde section du projet consistait à créer une base de données qui pourra contenir l'ensemble des données nécessaires autant pour la recherche que pour la récupération de données pour les utilisateurs. Une base de donnée à trois tables à alors été initialisée. Premièrement, la table `Oeuvres` contient l'ensemble des identifiants des oeuvres du fichier `JSON` et le nom de l'oeuvre. Comme les murales n'ont pas de titre, le nom du fichier de l'image lui correspondant à été attribué. Ainsi, lors de la création d'un nouveau fichier `JSON` au niveau du serveur, les oeuvres ne changeront pas mystérieusement d'identifiant si un des fichiers sources est modifié.

Deuxièmement, la table `Users` contient pour l'instant la clé primaire `UserName`

, `PassWord` et `LastLog`. Il sera possible d'ajouter de l'information supplémentaire sur les usagers si nécessaire plus tard.

Troisièmement, la table `Critics` contient en clé primaire les colonnes `UserName` lié à la même valeur de la table `Users`, `IDOeuvre` liée à la colonne `ID` de `Oeuvres` et `Date` de type `Unix TimeStamp`, et les colonnes `Comment`, `Note` et `ShareLink`. Avec ces colonnes en clé primaire, on assure de garder l'historique de toutes les modifications apportées par l'utilisateur.

Les tests initiaux ayant bien été effectués, aucune complication majeur ne s'est présentée. La seule problématique notable fut perçue lors des tests utilisateur, lors de la récupération des données, une incohérence avec les caractères spéciaux empêchait le bon visionnement des commentaires. Beaucoup de lecture sur la documentation à ce sujet s'est conclu par l'ajout d'une simple ligne précisant que la connexion entre les fichiers `PHP` et la base de données `MySql` se faisait en `UTF-8`.

La communication avec la base de données se fait via le serveur http://www-etud.iro.umontreal.ca/~beaurevg/ift3150/server/ où il est possible d'ajouter les paramètres `request`(`string`), `username` (`string`), `password`(`string`), `IDOeuvre`(`int`), `comment`(`string`), `note`(`int`) et `shareLink`(`string`). Les méthodes de requête `POST` et `GET` sont acceptées tant que seulement un des deux est exploité.

Une section de sécurité devra être implémentée à ce code afin de respecter éthiquement les données des utilisateurs. Il devra également y avoir une section pour sécuriser contre les attaques sur la base de données.

4.3 Récupération de données

Cette Section n'a pas encore été complété. Jusqu'à maintenant, la récupération de données se fait via un terminal connecté au serveur en ssh. Il est prévue que cette section soit complétée avant la présentation du projet qui se déroulera le 23 avril 2018.

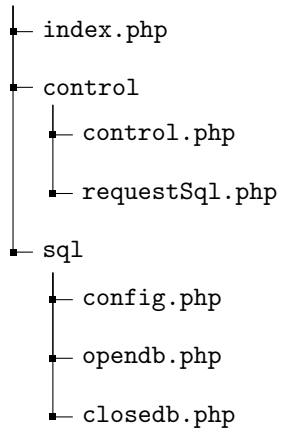
Cette tâche consiste à créer une page web qui devra permettre de récupérer les commentaires des différents utilisateurs pour chaque oeuvre et permettre un tri par date ou par utilisateur sans divulguer de renseignement personnel afin de récupérer les données dans le respect des utilisateurs.

5 Décomposition

Le fichier `index.php` s'occupe de diriger la requête initiale. S'il s'agit d'une requête concernant le fichier `JSON`, il l'envoie à `control.php`, sinon, il l'envoie à `requestSQL.php`. Dans le sous-répertoire `sql` se trouve l'ensemble des éléments

nécessaires afin de produire une connexion à la base de données `MySql`. Le fichier `config.php` est celui qui contient toutes les informations de valeur nécessaires à la connexion tels le nom du serveur qui héberge, le nom de la base de données, le nom de l'utilisateur et le mot de passe. `opendb.php` ouvre la connexion orientée objet et `closedb.php` la ferme. Dans le sous-répertoire `control` se trouvent tous les fichiers effectuant des calculs importants. On retrouve dans `control.php` l'ensemble des fonctions nécessaires à la création du fichier `JSON`, alors qu'on retrouve dans `requestSql.php` l'ensemble des fonctions nécessitants une interaction avec la base de données.

http://www-etud.iro.umontreal.ca/~beaurevg/ift3150/server/



6 Mode d'emploi

Afin d'accéder à toutes interactions avec les serveurs, le tout ce fait via l'url http://www-etud.iro.umontreal.ca/~beaurevg/ift3150/server/. Il est possible d'entrer les paramètres suivants sous format `POST` ou `GET` : `request`(`string`), `username` (`string`), `password`(`string`), `IDOeuvre`(`int`), `comment`(`string`), `note`(`int`) et `shareLink`(`string`). Voici les différentes fonctions possibles en attribuant les éléments suivants au paramètre `request` :

1. `createJson` : retourne le fichier `JSON` créé via la après analyse.
2. `loadJson` : retourne le fichier `JSON` dans le serveur
3. `createUser` : crée un nouvel utilisateur, nécessite les paramètres `username` et `password` valides.
4. `logUser` : retourne 1 si l'utilisateur existe, nécessite les paramètres `username` et `password`.
5. `setLastLog` : modifie la date de dernière connexion d'un utilisateur, nécessite les paramètres `username` et `password` valides.

6. `getLastLog` : retourne la date de la dernière connexion d'un utilisateur, nécessite les paramètres `username` et `password` valides.

7. `addComment` : ajoute un commentaire à une oeuvre selon un utilisateur. Nécessite les paramètres `username` , `password` et `IDOeuvre` valides et `comment`.

8. `addNote` : ajoute une note à une oeuvre selon un utilisateur. Nécessite les paramètres `username` , `password` et `IDOeuvre` valides et `note`.

9. `addShareLink` : ajoute un lien de partage à une oeuvre selon un utilisateur. Nécessite les paramètres `username` , `password` et `IDOeuvre` valide et `shareLink`.

10. `addCritic` : ajoute un lien de partage à une oeuvre selon un utilisateur. Nécessite les paramètres `username` , `password` et `IDOeuvre` valides et les paramètres optionnels `shareLink`, `comment` et `note`.

11. `getComment` : retourne le plus recent commentaire à une oeuvre par un utilisateur, nécessite les paramètres `username` , `password` et `IDOeuvre` valides

12. `getNote` : retourne la plus récente note à une oeuvre par un utilisateur, nécessite les paramètres `username` , `password` et `IDOeuvre` valides

13. `getShareLink` : retourne le plus recent lien de partage à une oeuvre par un utilisateur, nécessite les paramètres `username` , `password` et `IDOeuvre` valides

14. `getCritic` : retourne la plus récente critique à une oeuvre par un utilisateur sous la forme d'un fichier `JSON` avec un seul élément aux paramètres `IDOeuvre`, `comment` , `note` et `shareLink`. Nécessite les paramètres `username` , `password` et `IDOeuvre` valides

15. `getUserData` retourne toutes les critiques les plus récente d'un utilisateur pour chaque oeuvre sous forme d'un fichier `JSON` ou chaque élément est représenté comme les éléments retournés par la fonction `getCritic`. Nécessite les paramètres `username` et `password` valides

7 Conclusion

Bien que ce projet soit bien avancé, plusieurs éléments devront être complétés avant le lancement de l'application. Il faudra changer la technique de `logIn` d'un utilisateur afin de rendre la connexion sécuritaire et surtout difficile à pirater, il faudra prévoir d'autres champs dans la base de données concernant les utilisateurs afin d'optimiser la pertinence de la recherche. Bien que l'idée a été rejetée en début de session, les directeurs de recherche en histoire de l'art nous ont informés qu'ils désireraient récupérer les photos prises par les utilisateurs,

il faudra alors prévoir une section pour récupérer les images. Il faudra finalement optimiser l'analyse des `API` sources afin de supprimer les répétitions qui ont échappées aux précédents filtres.

La même équipe de programmeurs poursuivra le projet pendant l'été 2018 afin de se rendre jusqu'à la publication de l'application estimée peu avant le début de l'automne 2018. Un contrat de 50 heures sera signé pour compléter la partie serveur.

De ce projet, une bonne leçon peut en être tiré : "Sometimes it pays to stay in bed on Monday, rather than spending the rest of the week debugging Monday's code." - Christopher Thompson