GIOVANNANGELI Loann
RIOU Maxence
MOUHOUB Noureddine
CHEMOUNE Aleaddine

INF601 A1

Rapport
Projet de communication transdisciplinaire (PCT)

L3 Informatique
Université de BORDEAUX
2016/2017

Chef de projet : Loann GIOVANNANGELI

Table des matières

Introduction

La problématique proposée par le Musée des Beaux-Arts de Bordeaux est la suivant :

Le musée possède un grand nombre d'oeuvres et n'a donc pas la place de les exposer toutes en même temps. Il a donc prévu une salle pour entreposer des oeuvres qui étaient en réserve et désire que les oeuvres exposées dans cette salle alternent et que ce soit les visiteurs (ou visiteurs potentiels) qui choisissent ces oeuvres.

Pour répondre à cette problématique, nous avons réalisé une solution en deux applications qui permet au musée de créer un sondage et aux utilisateurs de voter pour une oeuvre dans ce sondage.

Les besoins sont spécifiés dans le cahier des charges correspondant (présent dans le git ci-dessous).

Dans ce rapport, nous présenterons le fonctionnement notre solution et expliquerons les choix que nous avons faits

Le code de ce projet est disponible sur le git :
https://github.com/LoannGio/PCT_Reserves

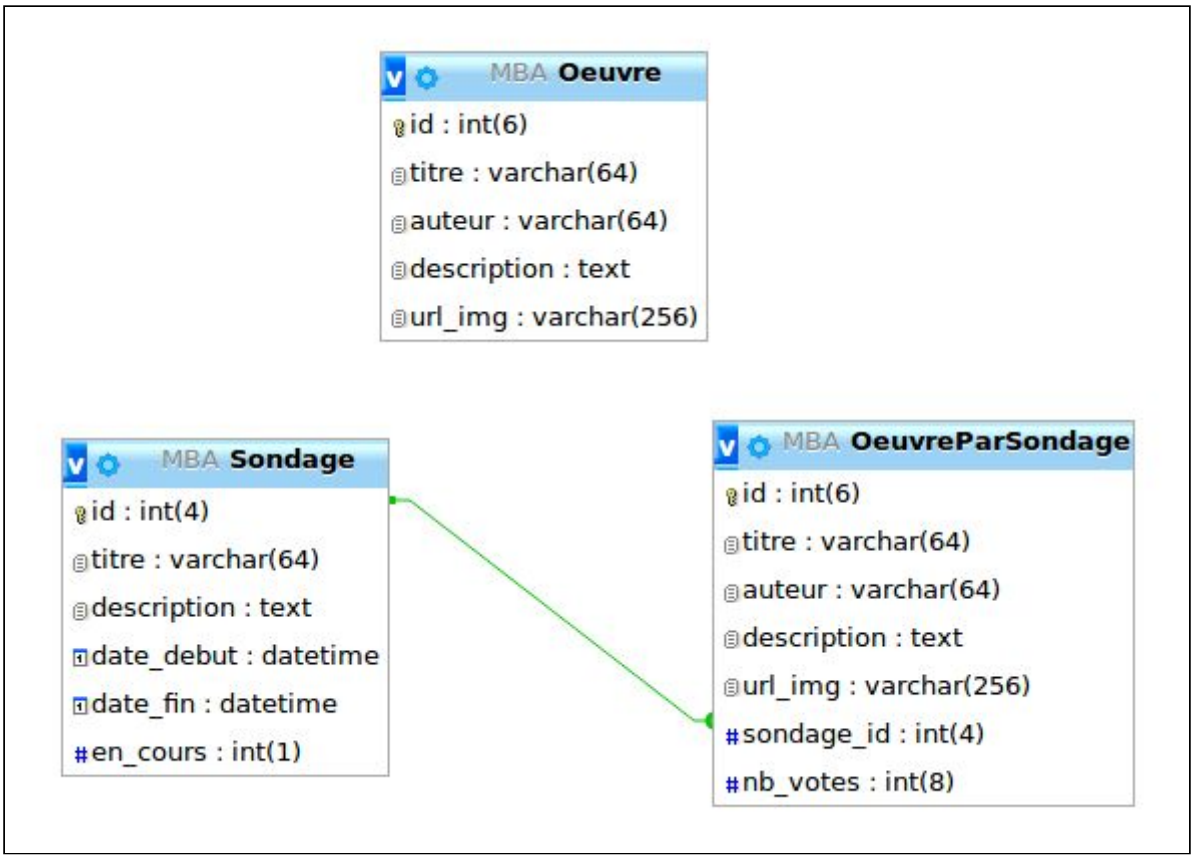
I - Base de données

Une base de données MySQL stocke toutes les données nécessaires au fonctionnement de nos 2 applications (sondages en cours, oeuvres d'un sondage, nombre de votes de chaque oeuvres, etc…). L'application interne écrit dans la base de données : ajout/suppression de sondages, ajout/suppression d'oeuvres. L'application web lit dans la base et écrit uniquement le nombre de vote d'une oeuvre lorsqu'un utilisateur réalise un vote.

La base de données comporte 3 tables :

- Oeuvre : toutes les oeuvres disponibles pour la création d'un sondage
- OeuvreParSondage : toutes les oeuvres d'un sondage identfiié par une clef étrangère
- Sondage : tous les sondages

Une oeuvre peut être instanciée plusieurs fois dans OeuvreParSondage si elle est présente dans plusieurs sondages. Une instance d'OeuvreParSondage possède les mêmes champs qu'une oeuvre et une clé étrangère qui correspond à l'**id** d'un sondage, ainsi qu'un champ **nb_votes** initialisé à 0.



II - Application musée locale

Cette application permet la gestion de la base de données qui stocke l'ensemble des informations nécessaires au bon fonctionnement du site web de vote.

Elle est locale par nécessité : utilisant une connexion à une base de données, la réaliser sous forme d'application web en ligne est certes plus facile d'utilisation, mais pose surtout des problèmes de sécurité vis-à-vis de la connexion en tant qu'administrateur et des droits/accès que cela peut procurer. En réalisant une application locale, seules les personnes à qui l'on a partagé cette application auront accès aux privilèges administrateurs (i.e. accès base de données).

Nous avons réalisé cette solution a été développée en C# Windows Form. Nous avons retenu ce langage car c'est un langage de production puissant qui comporte beaucoup d'outils de développement et de bibliothèques, la documentation mise en ligne par microsoft est exhaustive et il est facile de trouver des solutions/aides sur les forums. En outre, c'est un langage largement utilisé et qui a déjà fait ses preuves dans ce genre d'application locale.

Son plus grand défaut est qu'il n'est pas portable (notamment sous Linux) mais étant donné que près de 95% des ordinateurs utilisent windows, le risque que nous prenons est assez faible. La contrainte "l'application doit être compatible Linux" est donc forte et, si elle existe, aurait dû être précisée dans le cahier des charges.

Les fonctions que doit remplir cette application sont les suivantes :

- Posséder une page d'accueil présentant la liste de tous les sondages existants ou ayant existés
- Posséder une page type d'informations permettant d'afficher les informations complètes de n'importe quel sondage
- Permettre de créer un sondage, exporter ses informations à n'importe quel instant et pouvoir mettre fin à ce sondage
- Importer des oeuvres dans la base de données à partir d'un fichier XML (ou CSV dans une moindre mesure)
- Supprimer des oeuvres de la base de données

Les sous-parties suivantes présenteront comment ces fonctionnalités ont été implémentées fenêtre par fenêtre.

1) Accueil

Cette fenêtre est la fenêtre centrale de l'application : c'est en la fermant qu'on ferme l'application.

Elle comporte principalement une liste d'objets (scrollable) chargés à partir des sondages, terminés ou non, la base de données. On classe les sondages par ordre alphabétique en fonction de leur état (les "En cours" sont au dessus des "Terminé"). Chaque item de cette liste est composé des informations notables du sondage, à savoir : son titre, son état, ses dates de début et de fin annoncées.

N.B. les dates de début et de fin sont demandées et présentées à titre indicatif. En outre, il n'y a pas d'horloge qui viendra fermer automatiquement le sondage le jour de la date de fin annoncée de celui-ci.

La fenêtre d'Accueil comporte deux boutons de redirection : un vers la création d'un sondage et l'autre vers les pages de gestion de la base de données.

De plus, en double-cliquant sur un sondage dans la liste, on accède à sa page d'informations détaillées (permettant aussi l'export des informations du sondage et de mettre fin à celui-ci s'il est en cours).

2) Import

Cette page permet d'importer des oeuvres dans la table correspondante de notre base de données et possède un bouton deux boutons de redirection : "Retour" vers la fenêtre d'accueil et "Supprimer" vers la fenêtre de suppression

Pour ce faire, il suffit de cliquer sur le bouton "Importer". Un explorateur de fichier s'ouvre et il faut sélectionner un fichier ; extension ".xml" ou ".csv" obligatoire.
Ces fichiers doivent avoir un structure spécifique pour que l'import se déroule correctement.

Nous avons fait ce choix de fichiers d'imports car il est assez commun de pouvoir exporter les tables d'une base de données sous forme de fichiers de format xml, json ou csv.

Cet import supprime les anciennes oeuvres stockées dans la base, ce n'est pas simplement un ajout mais plutôt une réécriture.

a - Structure du fichier XML

```
<OEUVRES>
    <OEUVRE>
        <titre></titre>
        <auteur></auteur>
        <description></description>
        <url_img></url_img>
    </OEUVRE>
</OEUVRES>
```

Les balises XML :

<OEUVRES> au tout début </OEUVRES> à la fin du fichier. Représentent la liste d'oeuvres à ajouter dans la base de données.

<OEUVRE></OEUVRE> correspondent à une oeuvre, ces balises se trouvent toujours à l'intérieur des balises de la liste d'oeuvre.

<titre></titre>

<auteur></auteur>

<description></description>

<url_img></url_img>

On retrouve ces balises dans chaque balise <OEUVRE></OEUVRE>. Ce sont les informations relatives à une oeuvre.

Si cette structure n'est pas parfaitement respectée, l'importation échoue et un message d'erreur apparaît.

Remarque : les balises d'informations concernant une oeuvre peuvent être laissées vides.

b - Structure du fichier CSV

Exemple :

```
titre, auteur, description, url_img
"Le ciel bleu", "Henry PREVOT", "une peinture", "http://monsite.fr/img2.JPG"
```

La première ligne est obligatoire, elle présente les différents champs d'informations d'une oeuvre. Pour chaque oeuvre, ces champs sont obligatoires. Cependant ils peuvent être laissés vide.

3) Suppression

Cette fenêtre permet de supprimer une oeuvre de la base de données et contient un bouton de redirection “Retour” vers la page d’Import.

Elle contient principalement une grille d’oeuvres ou chaque ligne est une série d’informations sur une oeuvre (colonnes : image, titre, auteur, etc …)

Pour supprimer une oeuvre, on la sélectionne en cliquant sur la ligne correspondante dans la grille et on clique sur le bouton “Supprimer”.

4) Création

Cette page permet de créer un sondage et contient un bouton de redirection “Retour” vers la fenêtre d’Accueil.

Pour créer un sondage, il faut :

- renseigner un titre qui ne doit pas déjà exister parmi les sondages existants
- renseigner une description (non obligatoire)
- renseigner une date de début via le calendrier qui commence à la date courante
- renseigner une date de fin via le calendrier qui commence à la date de début + 1 jour et s’actualise à cette valeur si on avance trop la date de début afin d’empêcher d’avoir une date de fin antérieure à la date de début du sondage
- sélectionner (cocher) au moins deux oeuvres dans la grille qui renseigne leurs informations respectives

Pour finaliser la création, cliquer sur le bouton Créer.

Si toutes les conditions précédentes sont remplies, l’utilisateur est redirigé vers la fenêtre d’Accueil où apparaît le nouveau sondage dans la liste (quelle liste ? cf. 1) Accueil)

Si une conditions est pas ou est mal remplie, la création n’est pas finalisée est un pop-up apparaît, renseignant l’erreur commise

5) Information sondage

Cette fenêtre permet de consulter les informations sur un sondage (en cours ou terminé), d'exporter les informations à l'instant de la consultation, de mettre fin au sondage (s'il est en cours) et contient un bouton de redirection "Retour" vers la page d'Accueil.

a - Informations

L'affichage de la fenêtre permet de consulter, via des zones de texte et une grille, les informations concernant le sondage et toutes les oeuvres qu'il contient (informations de base des oeuvres + nombre de votes).

b - Terminer

Si le sondage est en cours, le bouton "Mettre fin" se trouve sur la fenêtre et permet de mettre fin au sondage (il n'apparaîtra plus aux utilisateurs du site web).

N.B. Cette est manuelle mais devrait être réalisée en adéquation avec la date de fin renseignée lors de la création du sondage afin de ne pas tromper l'utilisateur

$N.B._2$ Cette opération est définitive : un sondage terminé ne sera pas re-ouvert via l'application.

c - Exporter

En tout temps, le bouton "Exporter" est présent et permet d'exporter au format XML les informations sur le sondage et toutes les oeuvres qui le composent (avec leu nombre de vote).

Fonctionnement :

- Cliquez sur exporter : un explorateur de fichiers s'ouvre
- Sélectionnez un emplacement et un nom de fichier (à savoir que le fichier de sortie sera un fichier XML, il est donc préférable de lui donner l'extension ".xml")
- Enregistrez

III - Site web

Afin de réaliser les objectifs nous avons choisis d'utiliser le framework Laravel qui est basé sur le langage PHP que l'on trouvait adéquat à nos besoins et convenable aux exigences du projet avec son une architecture MVC (Modèles vues contrôleurs). De plus Laravel est très utilisé : on peut trouver une documentation exhaustive sur internet.

Pour faciliter la navigation à l'utilisateur nous avons mis une petite barre de menu pour que la page d'accueil et le site du musée des beaux arts soient accessibles dans toutes les pages (règle des trois clics : toute page du site doit être accessible en au plus trois clics), les pages principales sont détaillées ci-dessous:

1) Page d'accueil

Cette page est accessible par l'url: "domaine/ " c'est la page où on affiche tous les sondages en cours dans notre base de données, les informations sont affichées (titre, semi-description,date de début et date de fin) dans une table de type "Bootstrap Datatable".

Nous avons choisi d'utiliser une Bootstrap-Datatable car celle-ci implémente déjà un système de limite d'affichage d'objet par page, de pagination, fonction de tri des colonnes et (surtout) une barre de recherche dynamique.

Chaque ligne de cette table contient un lien vers la page qui définit un sondage.

2) Page de sondage

Pour accéder à la description d'un sondage et la listes des oeuvres de ce dernier il faut accéder à l'url (domaine/sondage/$id_sondage).

Cette page contient un lien vers la page de toutes les oeuvres.

3) Page des oeuvres par sondage

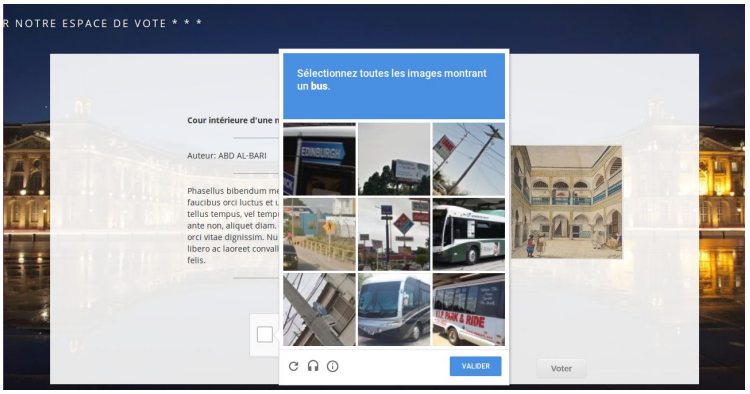
Cette page est accessible par l'url: "domaine/oeuvres/$id_sondage".
Elle affiche une datatable avec les informations de chaque oeuvre (image, nom, auteur, semi-description). avec la possibilité de chercher une oeuvre et de paginer.
Chaque ligne de la table possède un bouton pour accéder à la page d'oeuvre et de voter pour celle-ci.

4) Page oeuvre

L'url de cette page est: "domaine/oeuvre/$id_sondage/$id_oeuvre".
Elle affiche toutes les informations de l'oeuvre choisie avec une description complète et la possibilité d'agrandir l'image en glissant la souris vers l'image et le curseur de zoom apparaît.

En bas un bouton pour voter, avant que le vote soit possible il faut valider un captcha.

Si le captcha est validé, après le vote une page de remerciement apparaît pendant quelques secondes avant le retour à la page d'accueil, sinon un message d'erreur s'affiche.



5) Page d'erreur

Grâce à la puissance du framework utilisé précisément la gestion des exceptions qui nous a aider pour faire cette page importante.
cette page est appelé si l'utilisateur demande une url qui n'est pas définie sur le site.
Elle affiche le code de l'erreur déclenché (404, etc), avec un bouton de retour vers le site.

6) Design du site

Grâce à l'héritage des templates fournie par laravel nous avons créé une template générale qui possède une entête, un pied de page.
les autres pages du site héritent du template générale avec modification du contenu de la page (corps)
le style de la template générale est basé sur un thème de bootstrap.

Nous avons choisis Bootstrap pour définir le design intérieur de nos pages, et un design parmi plusieurs qui était convenable à notre site, alors on a pris les fichiers nécessaire pour établir le design avec quelques modifications apportées:

- On a mis une image d'arrière plan de la ville de bordeaux.
- Un diaporama de trois images comme présentation du musé.

Toutes les images utilisées par l'application sont sauvegarder dans un fichier img.

Pour bien adapté la datatable on était obligé de redéfinir certaines caractéristiques dans nos fichiers css et javascript comme l'emplacement des éléments de la page et la taille compatible.

IV - Organisation

Pour réaliser ce projet, nous nous avons commencés par créer la base de données : réfléchir à ce dont nous aurions besoins pour à la fois le site web et l'application locale, ce qu'il est probable que le musée contienne dans sa base de données, ce qui peut intéresser l'utilisateur… . Cette étape ne nécessite pas que personnes travaillent dessus en même temps, mais il nous semblait nécessaire que chacun puisse connaître la structure de la base de donnée afin de pouvoir l'installer et l'utiliser facilement par la suite.

Ensuite, nous nous sommes séparés en 2 groupes de 2 étudiants afin de travailler parallèlement sur le site web et l'application locale.

Nous n'avons pas travaillé en dehors des séances de Projet de Communication Transdisciplinaire (mis appart la rédaction de ce rapport) car nous n'en avions pas besoin : nous nous sommes imposés de venir au moins 2h toutes les semaines au cours de PCT, même lorsque ça n'était pas notre tour.

Cela permettait aussi d'avoir une entrevue "régulière" (hebdomadaire) avec le "client" (professeur). Ainsi, si nous prenions une mauvaise direction ou décisions,ou s'il y avait quelque chose à rectifier, nous ne perdions pas trop de temps (intérêt des méthodes agiles).

La méthode utilisée dans le développement de l'application web était une méthode incrémentale. L'objectif était d'avoir toutes les semaine un site entier et fonctionnel, et de ,chaque semaine, améliorer son ensemble.

Celle utilisée pour le développement de l'application locale était itérative : nous implémentions toutes les fonctionnalités d'une fenêtre, les testions, et une fois que nous étions satisfait : nous recommencions sur une autre fenêtre.

N.B. On parle ici de méthode itérative et incrémentale par abus de langage. Nous n'avons pas souhaité appliquer une méthode à la lettre mais il s'est trouvé, à posteriori, que nos méthodes se rapprochaient de celles ci.

Bilan

Nous sommes satisfait du travail réalisé. La quasi-totalité des fonctionnalités que nous souhaitions implémentées l'ont été. Bien sûr des améliorations peuvent être apportées :

- l'exécutable : celui que nous fournissons fonctionne lorsque les images affichées dans l'application ont une URL valide. Dans le cas contraire il va chercher une image stockée en local mais il ne la trouve pas lorsqu'on lance l'application via l'exécutable. Le chemin vers cette image n'est pas le même lorsque nous utilisons l'application en mode debug et lorsque nous la lançons via un exécutable.

- personnalisation de la connexion à la base de données. Si l'on souhaite utiliser une base de données différente, il faut modifier le code des deux applications. L'idéal aurait été de pouvoir faire cela depuis un fichier de configuration pour ne pas toucher au code.

- gestion des problèmes de syntaxe des fichiers d'import : lorsque la structure d'un fichier n'est pas conforme, l'utilisateur est prévenu. Cependant, il aurait pu être utile d'indiquer le type d'erreur ou bien son emplacement en donnant la ligne, le noeud, ou bien la colonne. Heureusement de nombreux outils en lignes sont disponibles et peuvent résoudre ce problème. Pour les trouver il suffit simplement de rechercher "csv validator" ou "xml validator" sur le web.

- statistiques : établir un classement des oeuvres en fonction de leur nombre de votes. Pouvoir connaître les oeuvres qui sont souvent bien classées à la fin d'un sondage.

- accueil : améliorer le design en ajoutant des styles, des couleurs.

- tableaux de sondages/oeuvres : ajouter une barre de recherche.

Conclusion

Ce projet nous aura permis de progresser en développement.

Les deux membres chargés de l'application web ont découvert le framework PHP Laravel : le routage, les contrôleurs, les vues. Ils ont également appris à insérer un re-captcha Google.

Un des deux autres membres sur l'application interne a dû réapprendre le C# ainsi que le "parsage" des fichiers XML et CSV dans ce langage. Nous avons également appris à mieux maîtriser Git, qui nous sera certainement très bénéfique lors de nos développements futurs.