L3 Informatique – Projet Annuel

Compte-rendu de projet

DALLOCCHIO François (**21702081**), DENOUAL Axel (**21600639**),
ROUSSEAU Alexy-Andréa (**21910036**), SABATIER Brian (**21907858**)

Table des matières

1. Introduction

Nous avons développé, au cours du dernier semestre de notre année de Licence 3 Informatique une application permettant de créer et de gérer une collection de photographies.

Celle-ci devait permettre à un conservateur de musée, ou à un amateur d'art de compiler et de proposer aux visiteurs de son site internet un parcours parmi des photos.

Cette application repose sur deux grosses briques logicielles :

- La première est un framework MVCR « maison » orienté objet, développé en PHP.

- La seconde est une application basée ReactJS, elle nous a permis permet d'obtenir relativement rapidement une interface de gestion d'images à la fois ergonomique et bien conçue.

➢ Ces deux composants logiciels ont été imbriqués l'un dans l'autre de sorte à ne former qu'un seul et même livrable.

Figure 1 - Une application ReactJS emboîtée dans une application PHP

Voyons à présent plus en détail comment s'est organisé le développement de notre galerie.

3. Organisation du développement

3.1 Utilisation de GitHub et communication de groupe

La célèbre forge en ligne a été employée; à la fois intuitive et complète elle a été une véritable aide pour permettre à chacun de suivre et contribuer au projet.

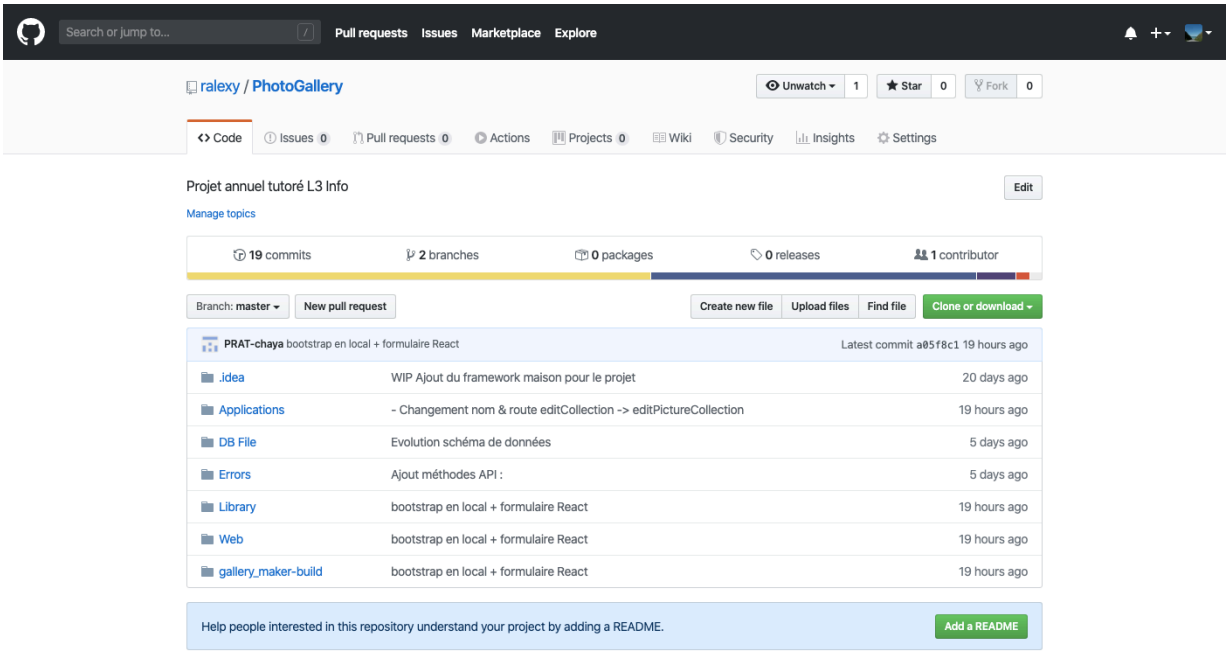


Figure 2 - Le projet sur la forge en ligne GitHub, remarquez les deux branches distinctes

Un groupe WhatsApp a été créé, il a été fort utile pour ne laisser personne pour compte et permettre d'inclure chaque membre de l'équipe activement dans le projet. Tout le monde pouvait communiquer à tout instant avec ses pairs afin de leur faire part de n'importe quelle remarque, question ou information relative au projet.

3.2 Séparation du travail en deux lots

Pour respecter une contrainte calendaire assez stricte il a été décidé de séparer nos efforts en deux équipes, ayant chacune travaillé sur une partie de la galerie photo.

- Chaque équipe a développé une « brique » logicielle majeure évoquée précédemment.

- Nous avons ainsi pu travailler simultanément sur le projet afin d'avancer dessus efficacement et bien plus rapidement.

3.3 Fusion du code de chacun

A l'issue de notre développement nous avons mis en commun notre travail

Le gros défi a été de faire en sorte de pouvoir faire communiquer aisément et le plus rapidement possible ces deux éléments complémentaires, cette problématique sera davantage développée en section 4.3.3.

4. Architecture du projet

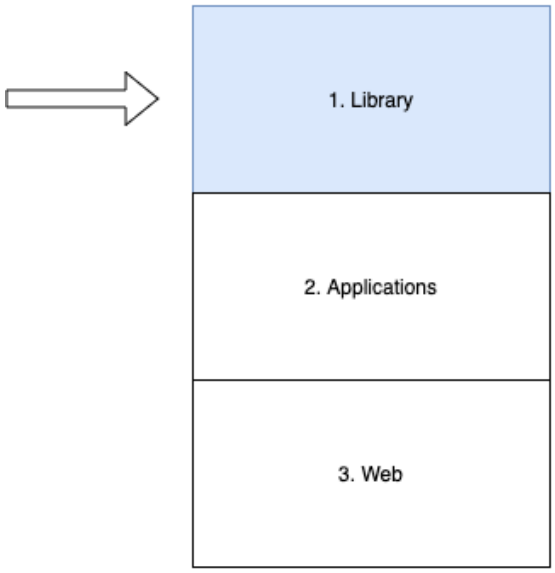
4.1 Généralités

Notre code a été extrêmement factorisé, hiérarchisé dans le but d'éviter des duplications de code inutiles et pour faciliter le réemploi de certains éléments de celui-ci.

Des illustrations seront employées tout au long de cette section afin de permettre une meilleure compréhension de notre travail tant il peut être ardu de l'appréhender sans aide visuelle.

Découvrons dès à présent étape par étape son fonctionnement.

4.2 La Librairie




```
rousseaualexy@Rousseaus-MacBook-Air Library % tree
.
├── Application.class.php
├── ApplicationComponent.class.php
├── BackController.class.php
├── Config.class.php
├── Entities
│   ├── Artist.class.php
│   ├── Collection.class.php
│   ├── Member.class.php
│   ├── Picture.class.php
│   └── PictureCollection.class.php
├── Entity.class.php
├── ExternalApi.class.php
├── HTMLMinifier.php
├── HTTPRequest.class.php
├── HTTPResponse.class.php
├── Manager.class.php
├── Managers.class.php
├── Models
│   ├── ArtistManager.class.php
│   ├── ArtistManager_PDO.class.php
│   ├── CollectionManager.class.php
│   ├── CollectionManager_PDO.class.php
│   ├── PictureCollectionManager.class.php
│   ├── PictureCollectionManager_PDO.class.php
│   ├── PictureManager.class.php
│   └── PictureManager_PDO.class.php
├── PDOFactory.class.php
├── Page.class.php
├── Pager.class.php
├── Route.class.php
├── Router.class.php
├── User.class.php
└── autoload.php

2 directories, 31 files
```

Figure 3 Le dossier Library

Le dossier Library contient le cœur du framework, c'est notamment grâce à ces classes que nous pouvons non exhaustivement :

- Gérer le routage de notre projet (Router.class.php),
- Gérer la connection ainsi que l'interaction avec la base de données (PDOFactory.class.php et Manager.class.php),
- Définir ce que doit être une Application de base (Application.class.php),
- Etc.

Il est intéressant ici de spécifier plus amplement l'utilisation ingénieuse des design pattern dans ce projet.

4.2.2 Singleton et Factory pour notre modèle

Le design pattern le plus simple utilisé dans cette application est Singleton.

- Il consiste à faire en sorte de ne pouvoir obtenir qu'une seule et même instance d'une même classe, rendant ainsi pour impossible la création de multiples objets issus d'une même classe.

➔ Nous l'avons utilisé pour l'objet PDOFactory, servant à la connexion à la base de données.

Factory consiste seulement à faire en sorte qu'un objet en instancie un autre.
Dans le cas présent notre classe PDOFactory instancie notre connexion à MySQL.

➔ Ainsi si nous voulons migrer à PostgreSQL, en utilisant toujours l'API PDO de PHP il nous suffirait de créer une méthode getPostgreSQL() pour y définir une connexion à ce SGBD.

4.2.3 Strategy pour notre modèle

Strategy a été utilisé dans notre dossier Models pour faire en sorte de pouvoir séparer les méthodes employées selon le SGBD utilisé.

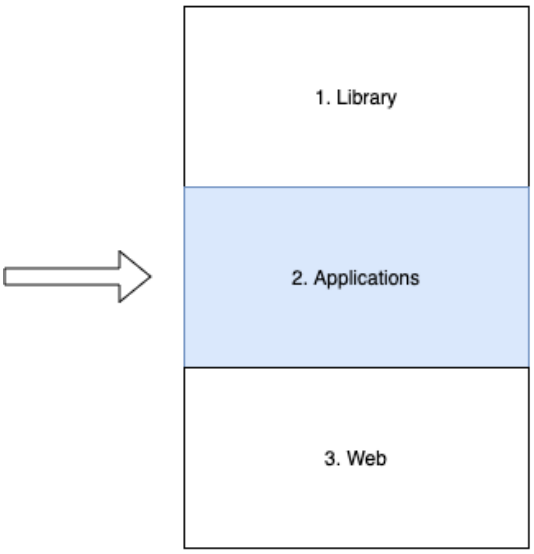
Plus simplement nous avons des classes du type :

- {NomEntite}Manager.class.php, abstraite définissant nos méthodes à écrire,
- {NomEntite}Manager_{Strategy}.class.php écrivant les méthodes propres à la stratégie choisie.

- La stratégie à adopter est directement injectée une et une seule fois à la construction de l'objet Managers.class.php, celui-ci en fonction de la stratégie fournie se chargera d'instancier les bonnes classes pour interagir avec le SGBD.

➔ Ces design pattern rendent donc aisé le changement de SGBD.

4.3 L'Application




```
rousseaualexy@Rousseaus-MacBook-Air Applications % tree -d
.
├── Api
│   ├── Config
│   ├── Lang
│   ├── Modules
│   │   ├── Index
│   │   │   └── Views
│   │   ├── Unmodule
│   │   │   └── Views
│   │   ├── addArtist
│   │   │   └── Views
│   │   ├── addCollection
│   │   │   └── Views
│   │   ├── addPicture
│   │   │   └── Views
│   │   └── editPictureCollection
│   │       └── Views
│   └── Templates
├── Backend
│   ├── Config
│   ├── Modules
│   │   ├── Index
│   │   │   └── Views
│   │   ├── Login
│   │   │   └── Views
│   │   └── Unmodule
│   │       └── Views
│   └── Templates
├── Frontend
│   ├── Config
│   ├── Lang
│   ├── Modules
│   │   ├── Index
│   │   │   └── Views
│   │   ├── PhotoShow
│   │   │   └── Views
│   │   └── Unmodule
│   │       └── Views
│   └── Templates
└── UneApplication
    ├── Config
    ├── Modules
    │   └── UnModule
    │       └── Views
    └── Template

44 directories
```

Figure 4 Le dossier Applications

Une application est composée de 3 dossiers eux même composés de sous-dossiers :

- /Templates/Layout.php, qui est le fichier PHP contenant le squelette de la vue de l'application,

- /Config comme son nom l'indique est le dossier contenant les fichiers de configuration de l'application, ainsi que ses routes (ce qui permet de faire fonctionner les URL),

- /Modules contenant les modules de l'application.
    - A chaque module est associé une sous-vue et un contrôleur.

➔ Retenons qu'à chaque route dans une application est lié un module qui contient une sous-vue. Ce fonctionnement permet de bien séparer le code derrière le traitement de chaque URL et de respecter le modèle MVC.

4.3.1 Le Frontend

Le Frontend correspond à la vue de l'espace public du site web, typiquement c'est l'application par défaut sur laquelle atterrissent les visiteurs de la galerie.

Il ne contient que deux modules pour ce projet, « Index » page d'accueil du projet et « PhotoShow » permettant d'afficher une collection sélectionnée.

4.3.2 Le Backend

Le Backend est en réalité l'espace privé de notre application, il sert à limiter l'accès aux seuls administrateurs authentifiés.

Ces identifiants sont stockés dans le fichier de configuration XML du Backend, étant donné la nature didactique de ce projet il n'a pas été jugé crucial de les chiffrer. Dans un cadre réel on aurait chiffré le login et le mot de passe de sorte à ce qu'il ne soit pas possible de les déterminer en lisant le fichier XML.

Notre Backend ne contient que deux modules : « Login » et « Index ».

4.3.3 L'API

L'API pour Application Protocol Interface permet de faire interagir notre application ReactJS avec notre code PHP.

- Des requêtes HTTP avec envoi optionnel de données POST au format JSON permettent d'interagir avec la base de données, pour obtenir et gérer les collections d'images.

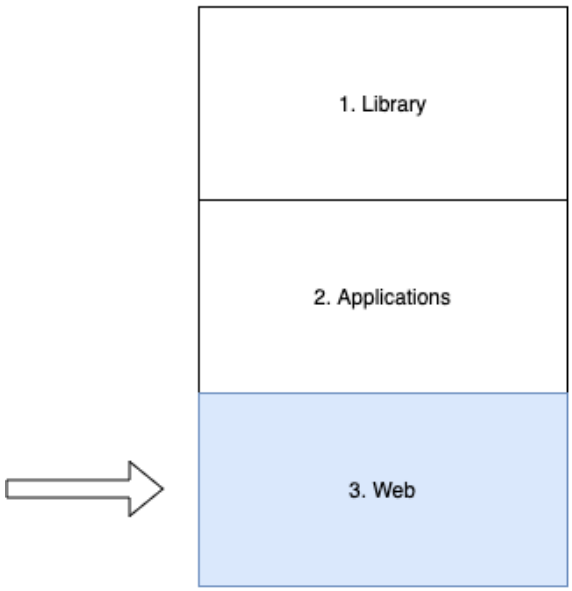
Quatre modules ont été créés, leurs noms étant assez explicites il est inutile de s'attarder sur leurs fonctions respectives :

- Index (listant toutes les images de toutes les collections),
- addArtist,
- addCollection,
- addPicture,
- addCollection.

Voici un exemple de retour de l'API, pour le module « Index » en l'occurrence :

[{"title":"Les demoiselles d'Avignon\r\n","year":"1907","artist":"Pablo Picasso","url":"http:\/\/photogallery\/pictures\/resources\/","thumbsUrl":"http:\/\/photogallery\/pictures\/resources\/thumbs\/","fileName":"1.jpg","pictureId":"1"},{"title":"Guernica","year":"1937","artist":"Pablo Picasso","url":"http:\/\/photogallery\/pictures\/resources\/","thumbsUrl":"http:\/\/photogallery\/pictures\/resources\/thumbs\/","fileName":"2.jpg","pictureId":"2"},{"title":"Peintre & Mod\u00e8le","year":"1928","artist":"Pablo Picasso","url":"http:\/\/photogallery\/pictures\/resources\/","thumbsUrl":"http:\/\/photogallery\/pictures\/resources\/thumbs\/","fileName":"3.jpg","pictureId":"3"},{"title":"Portrait de tante Pepa","year":"1896","artist":"Pablo Picasso","url":"http:\/\/photogallery\/pictures\/resources\/","thumbsUrl":"http:\/\/photogallery\/pictures\/resources\/thumbs\/","fileName":"4.jpg","pictureId":"4"},{"title":"La Joconde","year":"1503","artist":"L\u00e9onard De Vinci","url":"http:\/\/photogallery\/pictures\/resources\/","thumbsUrl":"http:\/\/photogallery\/pictures\/resources\/thumbs\/","fileName":"5.jpg","pictureId":"5"}]

4.4 Le dossier Web




```
rousseaualexy@Rousseaus-MacBook-Air Web % tree
.
├── api.php
├── backend.php
├── css
│   ├── Backend
│   │   └── style.css
│   └── sheet.css
├── fonts
│   ├── HelveticaNeueLight.woff
│   ├── photoshoot-webfont.woff
│   └── photoshoot-webfont.woff2
├── frontend.php
├── js
│   ├── gallery.js
│   └── jquery-3.4.1.min.js
└── pictures
    ├── icons
    │   ├── arrow-left.png
    │   ├── arrow-right.png
    │   ├── cross.png
    │   └── info.png
    └── resources
        ├── 1.jpg
        ├── 2.jpg
        ├── 3.jpg
        ├── 4.jpg
        ├── 5.jpg
        └── thumbs
            ├── 1.jpg
            ├── 2.jpg
            ├── 3.jpg
            ├── 4.jpg
            └── 5.jpg

8 directories, 24 files
```

Figure 5 Le dossier Web

Véritable point d'entrée de notre application c'est dans celui-ci que sont contenus nos fichiers PHP instanciant la bonne application.

Un .htpasswd permet de rediriger l'internaute vers la bonne application selon l'URL fournie :

- S'il va sur /admin/* il tombera sur le Backend,
- S'il va sur /api/{apiKey}/* il sera redirigé vers l'API,

Enfin s'il va sur toutes les autres adresses il sera redirigé par défaut sur le Frontend.

4.5 Le fonctionnement global du projet

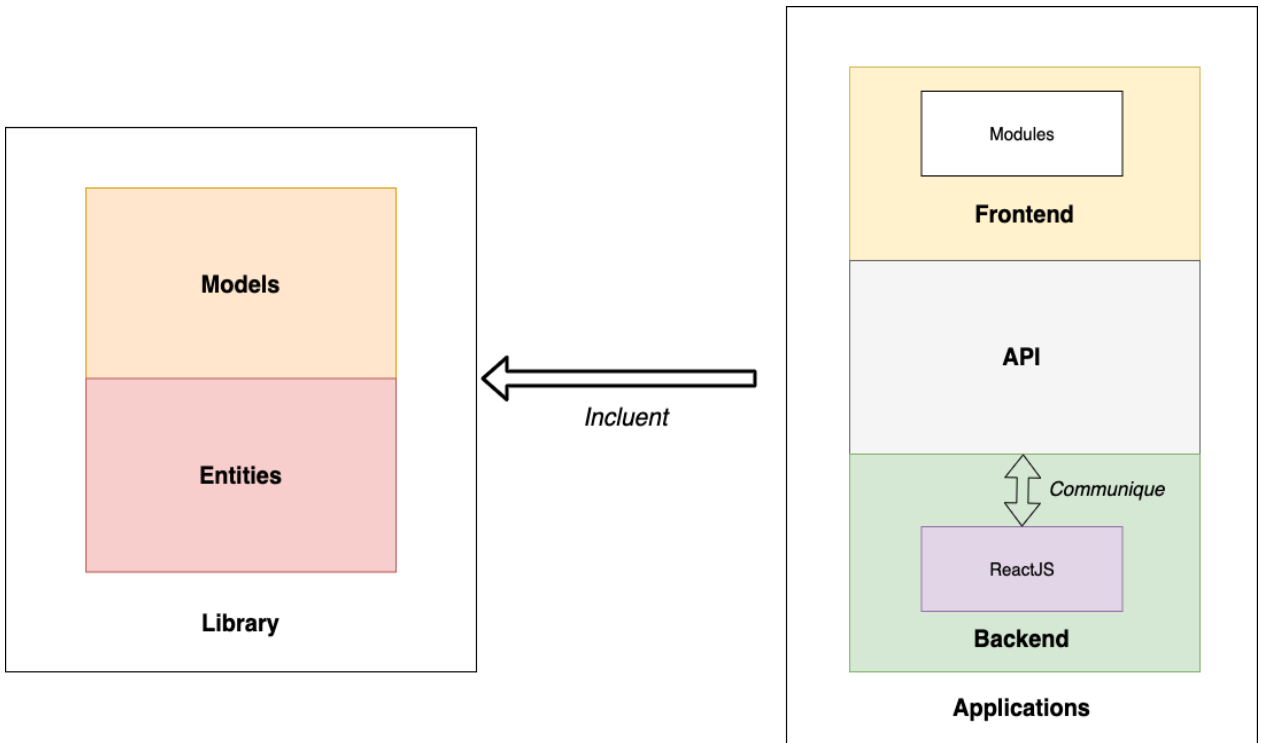


Figure 6 Un schéma est parfois plus parlant que plusieurs centaines de mots...

5. Petit aparté sur ReactJS

ReactJS permet avant tout de découper son application client en modules, qui sont ensuite importables et donc réutilisables à souhait.

S'ils héritent de la classe fournie « Component », on peut les invoquer dans une syntaxe JSX plus commode à lire et à utiliser que des manipulations directes du DOM comme on le fait en JavaScript classique.

En effet, on peut invoquer des composants au milieu du HTML en utilisant des balises superficiellement similaires à celles natives de HTML. React permet de passer des « props » à ces composants, en fait des propriétés pour le constructeur du composant.

Exemple de syntaxe : si je souhaite un panneau latéral qui contient des informations complémentaires, par exemple une description d'une image et sa date. En supposant que ces props sont définis correctement dans la classe du panneau latéral, je peux l'invoquer comme suit :

```
<div id="monAppli"> //...

<PanneauLateral description={"Très bonne description"}
date={"2019-02-06"} />

</div>
```

C'est ensuite la fonction de callback render (que tous les composants ont) qui va se charger de générer du HTML à partir de ces props. De plus, les composants possèdent une propriété « state » qui peut contenir une map clé/valeur qui correspond à des états que l'on souhaite garder dans tous les contextes tous en évitant de passer par des constantes (que l'on ne pourrait changer).

Une application écrite avec React doit être compilée vers du Javascript classique pour être utilisable par le navigateur. Heureusement, de nombreuses bibliothèques tel que Babel et Webpack facilitent cette tâche et permette même d'obtenir des fichiers avec du code minifié.

5 Fonctionnement de la galerie

5.1 La partie publique

Figure 7 La page d'accueil du site listant les collections créées, un simple clic mène à la collection.

Figure 8 Navigation dans la galerie via les boutons de contrôles

La navigation dans la galerie a été réalisée grâce à un petit script jQuery effectuant des requêtes XHR vers l'API qui retourne la liste des images.

Un clic sur les flèches de navigation mène aux images précédentes et suivantes, l'appui sur les touches précédentes et suivantes du clavier effectue les mêmes actions.

Un clic sur l'icône info en haut à droite masque les informations relatives à l'œuvre pour permettre une navigation plus agréable.

Un clic sur l'icône en haut à gauche permet de quitter la galerie et retourner à l'accueil, l'appui sur la touche echap produit le même effet.

5.2 La partie administrateur

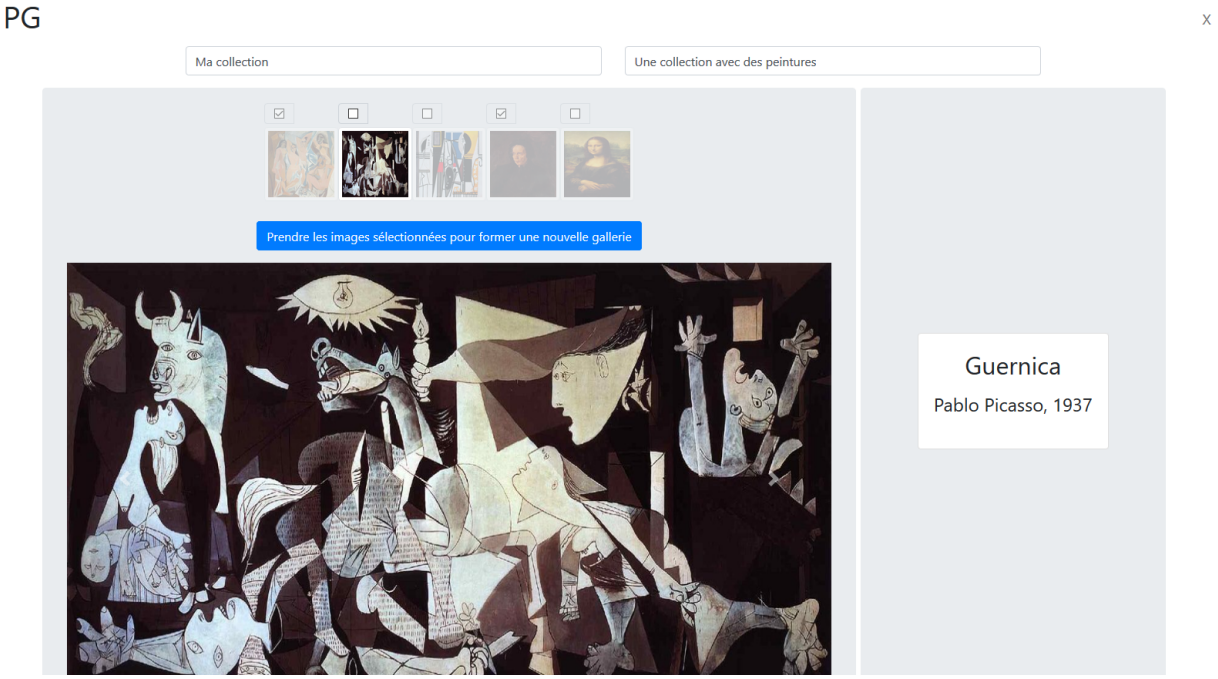


Figure 9 La partie administrateur du projet, permettant de gérer les collections

Identification du besoin : notre application de tri/création de galerie doit récupérer des images d'une base de donnée puis doit mettre à jour dans la base de données les tables qui gèrent les collections ordonnées.

Solution choisie : l'application ne renvoie les données au serveur que quand l'utilisateur sauvegarde. Le reste du temps, l'application doit garder l'état de la galerie.

Ainsi, l'application client récupère les données (en JSON) et renvoie en POST un JSON qui renseigne l'organisation de la nouvelle galerie.

Pour cela, elle est découpée en deux parties :

- La sélection des images parmi celles qui sont dans la base de données,
- Le rangement des images choisies.

5.2.2 Sélection des images :

La sélection des images se fait en cochant une checkbox collée à chaque image.

La liste d'images pouvant être grande, cette partie peut déborder à l'intérieur de son cadre (et on s'y déplace avec une barre de défilement).

On a une vue en détail de l'image si on la sélectionne simplement (pas besoin de cocher) ainsi que quelques informations comme le titre ou le nom de l'artiste.

Cocher une checkbox rajoute l'image à une liste temporaire qui est partagée avec le deuxième module de l'application.

5.2.3 Ordonnancement des images :

Le tri des images se fait par un simple glisser-déposer sur d'autres positions dans la galerie.

Chaque drag and drop « confirmé » change la liste temporaire.

On peut changer les informations liées à l'image si on le souhaite avec des formulaires sur le côté ou encore revenir à l'étape de sélection si on a oublié une image, sans perdre son avancement. Les images déjà sélectionnées seront déjà cochées.

Lorsque le tri est terminé, on doit rentrer un nom de galerie et une description pour celle-ci (accessible dans les deux étapes de l'application). Quand enfin l'utilisateur sauvegarde son travail, des requêtes sont envoyées pour sauvegarder cette galerie dans la base de donnée.

6 Points forts et limites

Ce projet a été une belle occasion de redécouvrir ce qu'était un travail en équipe. Nous avons décidé pour ne laisser personne en retrait et d'inclure aux maximum tous les participants à celui-ci, raison pour laquelle nous avons conçu un groupe WhatsApp.

Malheureusement vu le délai relativement court de réalisation de celui-ci et étant donné que nous devions le rendre durant une période d'examens certains membres de l'équipe, forcément moins rapides que d'autres ont dû se placer légèrement en retrait pour laisser les autres avancer bien plus rapidement.

De plus nous n'avons pas pu aller aussi loin que nous l'aurions voulu dans l'élaboration de la galerie, il aurait été très plaisant de rendre l'interface utilisateur pour les visiteurs bien plus ergonomique, esthétique, compatible.

Un mode de défilement automatique des images en continu était également prévu, il aurait permis de changer d'images toutes les 5 secondes par exemple.

Il aurait également été intéressant de créer une application mobile permettant de consulter et gérer nos collections grâce à l'API (comme nous l'avons déjà fait via le projet Android).

Enfin du temps supplémentaire nous aurait permis de rendre un meilleur rapport bien plus qualitatif, à la hauteur du travail qui a été fourni pour réaliser la galerie et son architecture.

7 Problèmes au déploiement

Le framework PHP sur lequel repose le projet nécessite de modifier la configuration du serveur web qui l'héberge en créant un VirtualHost.

Comme cette manipulation est impossible sur les serveurs de l'Université il a été nécessaire de louer un serveur pour y héberger l'application. En effet nous souhaitions éviter aux correcteurs de devoir effectuer une installation et une configuration de serveur chronophage pour tester notre application.

Il aurait été appréciable de disposer par exemple d'une machine virtuelle fournie par l'université configurable et utilisable en réseau pour de tels projets.

8 Pour conclure

Nous tenons à remercier tout particulièrement les enseignants qui nous ont appris cette année à réaliser des applications web utilisant l'architecture MVCR et nous ont initié à l'utilisation d'API JSON.

Enfin nous espérons que vous apprécierez notre réalisation et que vous prendrez en compte la forte contrainte temporelle ayant nécessairement nui à l'élaboration d'une solution bien plus proche de la perfection.