Rapport Projet Base De Données

Dumange Nicolas, Jeantet Gabriel

April 2021

Implémentation et modifications

Nous avons choisi d'implémenter notre base de donnée via PostgreSQL et PHP. Nous utilisons le serveur lighttpd avec l'extension php-cgi.

De plus, les relations de type N/N telles que la participation d'un artiste à un courant artistique sont représentées par des tables, permettant de rendre compte d'un artiste à cheval sur plusieurs courants ou d'un courant représenté par plusieurs artistes. Nous avons aussi inversés le sens de certaines relations 1/N, telles que la relation entre un mécène et un don, permettant d'intégrer le donateur dans le don, et de ne pas avoir à créer de table pour rendre compte de cette relation. Les mécènes ont été supprimés et sont désormais simplement intégré à la table Users. Nous avons également retiré le champ "don mensuel", car il ne semblait plus nécessaire. Nous avons également retiré l'id pour les artistes, et désormais la clé primaire est le nom de l'artiste.

Les héritages en SQL ne permettant pas de transmettre les clés primaires et ajoutant des complications, nous avons donc créé quatre tables distinctes exposition_temporaire_artiste, exposition_temporaire_courant, exposition_temporaire_pays, exposition_permanente, et nous avons crée une VIEW permettant de traiter toutes les expositions sans prendre en compte les champs supplémentaires des expositions temporaires. Afin d'assurer l'unicité des id dans cette VIEW, nous avons crée une table Expo_ids, qui nous permet de générer un id pour chaque exposition. Cela est assez contraignant car avant chaque insertion dans une table d'exposition il est nécessaire de créer manuellement un nouvel id.

Fonctionnalités

Les utilisateurs doivent se connecter pour accéder aux fonctionnalités. Il peuvent pour cela se créer un compte. Le site permet d'effectuer des recherches sur les oeuvres d'art, les artistes, les courants artistiques et les musées. Les utilisateurs peuvent effectuer des dons à un musée, accessible depuis la page d'accueil. Nous nous sommes arrangés pour que si l'utilisateur n'est pas connecté, il soit toujours redirigé vers la page de connexion. Les utilisateurs ont également deux champs supplémentaire, déterminant leur accès au panneau d'administration ainsi qu'à la gestion d'un musée. Un administrateur peut donner le rôle d'administrateur

à d'autres utilisateurs, et peut également attribuer la gestion d'un musée à un autre utilisateur. Il est donc nécessaire d'avoir un administrateur initial. Cependant nous hashons les mots de passe avant de les ajouter dans la table Users. Ainsi il est nécessaire d'ajouter le rôle d'administrateur initial manuellement dans la base de données. Un administrateur peut donc ajouter ou retirer le rôle d'administrateur à un autre utilisateur, et peut faire de même pour la gestion d'un musée. Lorsqu'un utilisateur gère un musée il accès à l'onglet management. Celui ci permet d'ajouter des oeuvres appartenant au musée, ou de déclarer de nouvelles expositions organisées par le musée. Enfin il peut également mettre à jour les données concernant une oeuvre en spécifiant l'id de cette oeuvre. Nous gérons ces rôles au moment de la connexion de l'utilisateur, nous spécifions alors certain champs lors de la création de la session ce qui permet de savoir tout au long de la session les rôles d'un utilisateur. Ainsi si le rôle d'un utilisateur est modifié, il devra se déconnecter et se reconnecter afin que son rôle soit mis à jour.

Détails sur la recherche

Lorsqu'un utilisateur fait une recherche, nous effectuons une requête et nous utilisons l'extension pg_trgm qui permet d'utiliser la fonction SIMILARITY, qui nous permet de trouver des résultats de recherche cohérent même si l'utilisateur a fait une faute dans sa recherche. Nous cherchons également les oeuvres qui ont le mot-clé en substring d'une colonne intéressante. Ainsi si l'utilisateur cherche "Le", il obtiendra notamment toutes les oeuvres dont le nom commence par "Le".

Modifications par rapport à la spécification

Difficultés rencontrées

Nous avons pris du retard sur le projet et n'avons pas pu implémenter la recherche avancée à temps. Aussi nous l'avons retiré de la barre de navigation. Cependant le fichier "advanced-search.php" est toujours présent bien qu'incomplet.

Structure du projet et manuel

Le projet est structuré de la manière suivante:

- un Makefile, comprenant les commandes nécessaires pour lancer les différents composants du projet, ainsi qu'une commande pour afficher et effacer les logs du serveur lighttpd.
- un dossier **www**, contenant tout le code php de l'application web.

- un fichier **install.sh**, qui n'est pas à exécuter mais qui indique les commandes nécessaires à effectuer afin que le projet fonctionne correctement. Cela comprend initialiser la base de donnée au bon endroit, créer des "softlinks" vers les différents fichiers de configuration de php, et de lighttpd, qui sont contenu dans le dossier config.

- un dossier **config**, contenant les fichiers de configuration de php et de lighttpd.

- un dossier **db**, contenant les scripts sql permettant de créer la base de données et de la remplir de données de test. La base de données doit s'appeler "arts", ainsi on peut la créer avec deux commandes "initdb arts", et "psql arts ¡ create_arts.sql". Il faut également un utilisateur "app" de mot de passe "app", qui est l'utilisateur que l'application utilise pour manipuler la base de données. Cet utilisateur est normalement crée lors de l'utilisation de create_arts.sql.

Php et lighttpd ont été installés localement pour notre projet, et nous utilisons le Makefile pour lancer l'application. Pour ce faire il faut utiliser trois commandes:

- make rundb. Qui permet de lancer la base de données.

- make runcgi. Qui permet de lancer le serveur cgi pour php.

- make runserv. Qui permet de lancer lighttpd.

Une fois ces trois commandes lancées, l'application est accessible en "localhost:8080". Initialement aucun utilisateur du site n'existe dans la base de données, les mots de passe étant hashés nous ne pouvons par les ajouter par une requête sql. **Dans le Makefile, et lighttpd.conf il est important de modifier les chemins afin qu'ils soit cohérents avec votre système de fichier (cela revient à changer le home). En cas de problème pour la configuration de l'application, ne pas hésiter à nous contacter. Le code est également disponible sur github : https://github.com/delkhos/projbdd .**