Projet DataWarehouseMining pour les archives départementales de Vendée

Binome

- Hugo Moracchini
- Anthony Lozano

Solution choisie

Pour ces travaux nous avons choisi de créer une application web, en effet les interfaces graphiques sont très simples a gérer et d'un point de vue comptatibilité et déploiement les solutions webs ont leurs avantages ...

Familiés avec PHP orienté objet et le développement web, nous avons choisi un assortiment d'outils de développement afin de s'exempter des tâches les plus pénibles et chronophages compte tenu des délais très courts. C'est pourquoi nous utilisons Laravel, qui fonctionne sur une architecture 'pseudo' MVC, il est très souple et l'ORM fournie (Eloquent) avec est simple à utiliser.

Nous avions prévu un certains nombre de fonctionnalités pour cette petite application mais malheurement nous avons manqué de temps.

Ce qui était prévu (avec ambition) :

- Tableau de bord
  - Gestion utilisateurs
    - Connexions et gestion des droits
  - Import de fichiers CSV
  - Traitement des données brutes (fichiers CSV) pour insertions dans la base
  - Gestion des données
    - CRUD disponible sur toute les entités de la base
    - Formulaire d'ajout d'actes
    - Outil de correction de données incohérentes et incorrectes
  - Exploitation des données
    - Vue générale
    - Visualisation avancées des données des tables
      - Graphiques
      - Maximums
      - Minimums
      - Moyennes ...

Ce qui a été fait :

- Tableau de bord
  - Traitement des donnèes brutes (fichiers CSV) pour insertion dans la base
  - Gestion des données
    - CRUD disponible sur la table `personnes`
  - Exploitation des données
    - Vue générale avec 1 graphique

Composition du dépôt

- LozanoAnthony-MoracchiniHugo-CR.[md|html|pdf]

    - Compte-rendu lui-même
- vendee.db.sql
    - Export de la base de données
- datamining-vendee.zip
    - Archive de l'application

Installation de l'application

Prérequis

- PHP fonctionnant en ligne de commande
- Composer
- Serveur Apache/MySQL/PHP

Installer la base de données grâce au script `vendee.db.sql`. En raison de sa taille (~170MiB) le plus simple est de lancer la commande suivante sur un shell SQL

```
source path/to/file.sql;
```

Récupérer le projet via l'archive dans le dépôt moodle ou cloner le projet avec git. Se placer dans le répertoire de l'application via ligne de commande et lancer la commande *composer* pour mettre à jour les dépendances.

```
git clone git@gitlab.com:anthozano/datawarehouseming-vendee.git
```

```
composer update
```

Créer un fichier de configuration pour Laravel :

```
cp .env.example .env
```

Et fournir les informations suivantes :

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=
DB_USERNAME=
DB_PASSWORD=
```

Configurer le framework Laravel

```
php artisan key:generate
```

Et enfin il faut configurer le serveur web pour que la racine du site web pointe sur le dossier `public/`.

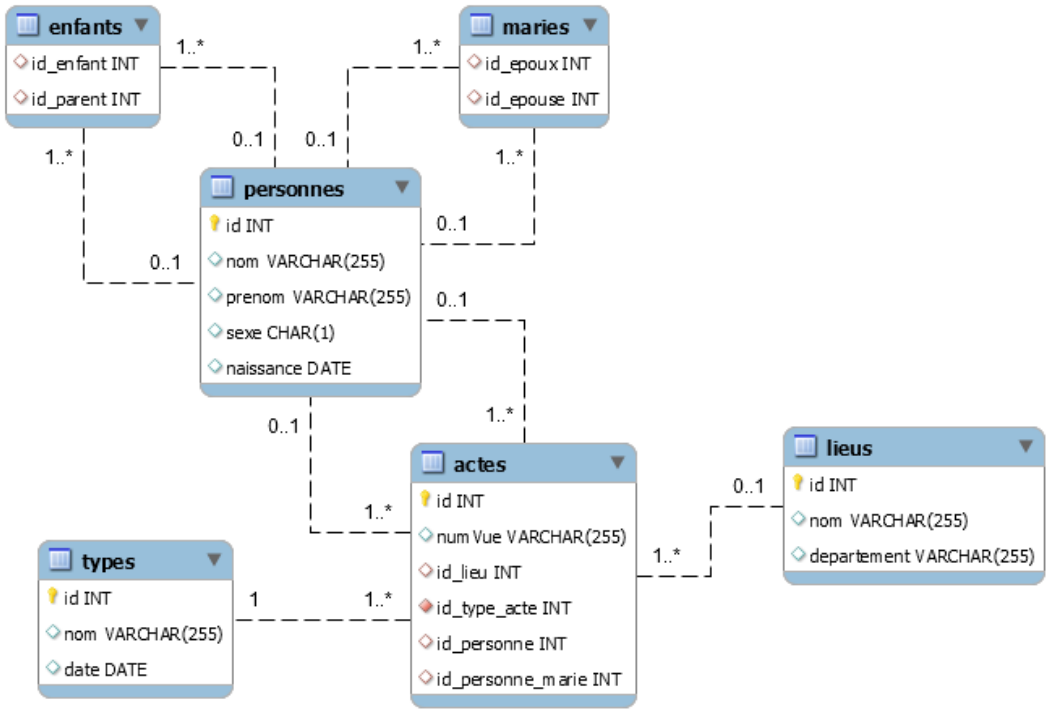
Outils utilisés

- Versionning : GIT avec le projet héberger sur gitlab.com
- Plateforme de développement (Apache/MySQL/PHP: WampServer
- Composer : Gestionnaire de dépendances PHP
- Framework Backend (PHP) : Laravel
- Framework Frontend (HTML/CSS/JS) : Bootstrap
- Plugin JS pour les graphiques : Chart.js

- Conception de la base de données : MySQL Workbench
- IDE : PhpStorm

Conception de la base de données

Nous avons créer le diagramme de conception de la base de données sur MySQL Workbench, grâce au reverse ingeneering le script de création de la base à été créer automatiquement.



La table `enfants` et `maries` représente respectivement les relations de parentés et du mariage. La relation `types` représente les types d'actes, la colonne `types`.`date` représente la date de mariage pour un acte de mariage et la date de décès pour un acte de décès. Presque toute les valeurs peuvent être nulle en raison des données sources qui sont trop approximatives pour pouvoir utiliser une base de données plus stricte au niveau de ses données. La seule ambiguïté que nous voyons dans cette proposition de base est la façon dont est géré la relation entre la table `actes` et `personnes`, la colonne `id_personne` correspond à la personne concernée et la colonne `id_personne_maries` représente la 2ème personne concerné par cet acte, dans le cas d'un mariage. Dans le cas d'un deces, la colone `id_personne_marie` prend la valeur NULL. La syntaxe au pluriel et simpliste des tables se justifie pour simplifier l'utilisation de l'ORM de Laravel. Ne sont pas présent sur le diagramme les 2 tables "brutes" qui correspondent aux fichiers CSV, elles ont été obtenu à partir d'un import faite avec une requête SQL.

Peupler la base

Pour peupler la base, il faut passer par l'interface web et aller dans la rubrique "Traitement des données brutes" et lancer le traitement. En raison du trop grand nombre de lignes à traiter, le script s'arrête car trop long à exécuter. Ce problème pourrait être éviter grâce aux **chunck** d'**Eloquent** mais nous n'avons pas eu le temps de travailler ce problème. Comme solution temporaires nous nous sommes contenté des 1000 première lignes de chaque fichiers CSV. (voir `App\Controller\DashboardController@processImport`).

Conclusion

Le traitements des données existantes était très difficile, il s'agissait de fichiers énormes avec des données de très mauvaises qualité, il a été difficile de typé toute ces données en particuliter pour les dates, qui était fournies sous forme d'age pour certaine. Malgré les nombreuses attentes sur le produit final et les délais extrêmement courts, ce projet était concret et représentait un challenge intérressant. Il nous a également permis de manipuler et de mettre en oeuvre beaucoup de technologie différentes.