Projet Réservations

Nom Prénom

Section, Année, École

Date

# 

# But du projet

Le projet consiste à informatiser la gestion des réservations de spectacles d’une société de production. Celle-ci gère un catalogue reprenant des spectacles, leurs auteurs et leurs metteurs en scènes, les comédiens, ainsi que les lieux et les dates de représentations.

**L’internaute** pourra consulter librement le catalogue des spectacles affichant le lieu et les prochaines dates de représentation. Il pourra effectuer des recherches, des tris et des filtres à travers les pages du catalogue.

**Le membre** pourra réserver des places pour une représentation d’un spectacle, consulter la liste de ses réservations et modifier ses données de profil.

**L’administrateur** pourra gérer son catalogue à travers un back-office sécurisé. Par exemple, il pourra ajouter, modifier et supprimer un spectacle manuellement, importer/exporter des données au format CSV, mais aussi mettre à jour la liste des spectacles grâce aux nouveautés publiées par un Web service tiers.

À son tour, **l’application** devra produire d’une part son propre Web service (une API authentifiée avec système d’affiliation), d’autre part un flux RSS (par exemple, la liste des prochaines représentations).

# 

# Vue Globale

Pour la création de ce projet, les technologies suivantes ont été utilisées:

* Langages: PHP (back-end), Javascript (front-end)
* Framework: Laravel + Extensions (Fast-Excel pour le CSV, Laravel Feed pour le flux RSS)
* Librairies front-end: JQuery, DataTables,
* Framework CSS: Bootstrap 4
* Système de template: Blade

Voici l’ordre des différentes étapes pour la création du site:

1. Installation et configuration des librairies et frameworks
2. Ajout des fichiers de langue, pour que Laravel affiche tous ses messages (ceux d’erreur par exempl) en français plutôt qu’en anglais
3. Création des modèles/migrations/seeder/controller/views pour la plupart des composantes du site (Artiste, Représentation, Show, Réservation, …)
4. Ajout du flux RSS
5. Ajout de l’import/export CSV
6. Ajout du service Rest

# Création d’une section du site

La création de la plupart des sections du site est faite de façon similaire, qui consiste en 5 étapes:

1. Création du modèle
2. Création de la migration pour ce modèle
3. Création du seeder pour ce modèle
4. Création du controller et des vues pour voir/créer/modifier/supprimer des instances du modèle
5. Ajout des nouvelles routes

Voici un exemple avec le modèle Artiste:

## Création du modèle

Pour créer le modèle Artist, nous devons préciser quels sont les attributs du modèle, le nom de la table, si on désire des timestamps et quelle sont les relations avec d’autres modèles (exemple: un spectacle peut avoir plusieurs représentations).

La commande *php artisan make:model* de Laravel permet de facilement générer une ébauche de Modèle, que nous personnalisons après. Dans le cas d’Artiste, nous utilisons les attributs *firstname* et *lastname* et comme nom de table *artists*.

## 2. Création de la migration pour ce modèle

Une fois que le modèle est créé, il faut créer la migration pour Artist, qui est responsable d’ajouter/enlever la table représentant les artistes dans la base de données.

La commande *php artisan make:migration* de Laravel permet de facilement générer cette migration (Laravel détecte automatiquement le nouveau modèle et crée une migration). Néanmoins, il est souvent nécessaire de modifier cette migration pour:

1. **Ajouter des contraintes (taille maximum):** par exemple, nous devons la préciser la taille maximum de colonnes *firstname* et *lastname* de la table *artist*
2. **Préciser si certains attributs peut être null:** par exemple, l’attribut poster\_url de la table *shows* peut être nulle, vu qu’il n’y a pas toujours de poster
3. **Ajouter des clés étrangères:** par exemple, nous précisons que la colonne *artist\_id* de la table *artist\_type* références la colonne *id* de la table *artist*

## 3. Création du seeder pour ce modèle

Il est utile d’avoir des données test lorsqu’on développe le site, pour mieux l’utiliser et le tester. Dans notre cas, on souhaite ajouter des exemples d’artistes, de spectacles, de représentations, … dans la base de données. Pour cela, nous créons un Seeder, qui est responsable de créer quelques exemples du modèle et de le sauver dans la base de données.

Dans le cas de Artist, nous créons un ArtistSeeder, qui ajoute les exemples fournis dans le Roadmap dans la base de données.

## 4. Création du controller et des vues pour voir / créer / modifier / supprimer des instances du modèle

Nous avons maintenant le modèle Artiste, la table correspondante dans la base de données et des exemple d’artistes. L’étape suivante est de pouvoir voir / créer / modifier / supprimer des artistes.

Pour cela, nous allons créer un controller (ArtistController) et plusieurs views (index, show, create, edit, delete) pour les différentes actions.

Le controller soit:

* se charge d’aller chercher les informations nécessaire sur l’artiste(s) et d’afficher la page demandée (exemple: afficher la liste des artistes).
* fait l’action demandée (la suppression par exemple), puis redirige l’utilisateur vers une autre page (la page avec la liste des artistes par exemple).

Le controller gère aussi les accès et les permissions. Il peut ainsi interdire certaines actions:

* Certaines actions ne peuvent être faites que par l’administrateur, comme par exemple supprimer ou modifier un artiste
* Certaines données ne peuvent être visibles que pour un seul utilisateur. Par exemple, le ReservationController interdit à un utilisateur de voir les réservations d’un autre utilisateur.

**Structure**

Nous nous sommes basés sur un exemple de qualité pour créer le controller et les views, ce qui nous a permis d’être sûr de gérer tous les cas (dont les erreurs) et de garder la même structure pour tous les controllers et vues.

## 5. Ajout des nouvelles routes

La dernière étape est d’ajouter les routes. Ainsi, on va par exemple indiquer que lorsqu’un utilisateur visite le lien /artist/, c’est la méthode index de ArtistController qui va générer le contenu à afficher. On fait ceci pour toutes les actions: voir la liste les artistes, voir les détails d’un artiste, créer un artiste, modifier un artiste et supprimer un artiste.

--------------------------------------------------------------------------------------------------------------------------------

En faisant ces 5 étapes pour tous les modèles et avec un peu de travail additionnel nous pouvons faire le use case

* Modifier son profil
* Réserver des places
* Consulter ses réservations
* Consulter le catalogue paginé des spectacles
* CRUD catalogue

Pour gérer les use cases Filtrer les spectacles, Trier les spectacles et Rechercher un spectacle, nous ajoutons DataTables à la page qui liste les spectacles.

# Se connecter / S’inscrire

Pour gérer ce use case, nous avons utilisé la commande *php artisan make:auth* de Laravel, qui génère un controller et des vues pour s’inscrire, se connecter et pour pouvoir modifier son mot de passe.

Bien sûr, nous avons dû personnaliser les controllers et les vues générées, car elles ne correspondent pas tout à fait à notre projet. Par exemple, nous avons dû ajouter la gestion du champ Prénom, Nom et Langue, ainsi que la connexion soit par login ou par email.

# Flux RSS

Pour générer le flux RSS des spectacles facilement, nous utilisons la librairie Laravel Feed. Il suffit de

* Définir quel modèle doit être utilisé (Show dans notre cas)
* Définir quelles instances du modèles doivent être transformées en items RSS (toutes dans notre cas)
* Définir comment transformer une instance du modèle en item RSS
* Définir l’URL du flux

et Laravel Feed se charge du reste. Après, il suffit d’ajouter un lien vers notre flux RSS dans la page qui liste les spectacles.

# Import/Export CSV

Pour importer et exporter des fichiers CSV, nous utilisons la librairie Fast Excel pour Laravel. Cette librairie permet, en quelques lignes de codes seulement, d’exporter et d’importer des instances d’un modèle donné. Nous avons aussi ajouté un petit formulaire (dans la page qui liste tous les spectacles) visible seulement pour l’administrateur, qui lui permet d’utiliser cette fonctionnalité.

# Restful API

Le site obéit à un REST API, vu qu’il permet de faire des opération de création, édition, listage, suppression et vue des détails pour les modèles importants en suivant les conventions REST. Les différents controllers renvoient une réponse JSON si ‘?json’ est ajouté à l’URL, permettant ainsi à des ordinateurs de facilement utiliser les résultats.

On utilise l’authentification par défaut, avec des cookies, par simplicité et limitation de temps.