COMPTE RENDU PROJET

INFO0303

Antoine Duval
Antoine Théologien
(S3F3)

Table des matières

I. Introduction

A. Sujet

Pour ce projet en langage PHP, de la matière INF0303, nous avons choisi le sujet 1 :

Une commune désire mettre en place une application pour gérer les activités périscolaires des enfants de ses administrés. Ces activités peuvent être la garderie (matin et soir pour les lundis, mardis, jeudis et vendredis) ou sur la journée (ou demi-journée) les mercredis.

Chaque créneau doit pouvoir être configuré avec une description. Chaque famille concernée doit être adhérente à l'association gérante de ces activités. Le tuteur légal peut inscrire son/ses enfants aux différentes activités proposées. Les inscriptions doivent être effectuées au minimum une semaine avant.

Un bilan est accessible pour chaque famille (liste de toutes les activités suivies par ses enfants). Les gestionnaires ont pour rôle de gérer les familles et les créneaux des activités. Ils peuvent réserver des créneaux pour des enfants et récupérer les bilans de chaque famille. L'application doit être fonctionnelle pour plusieurs communes.

Variante : Ce type de système peut également être utilisé pour gérer des activités hors périodes scolaires (comme le Reims Festival Été).

B. But du projet

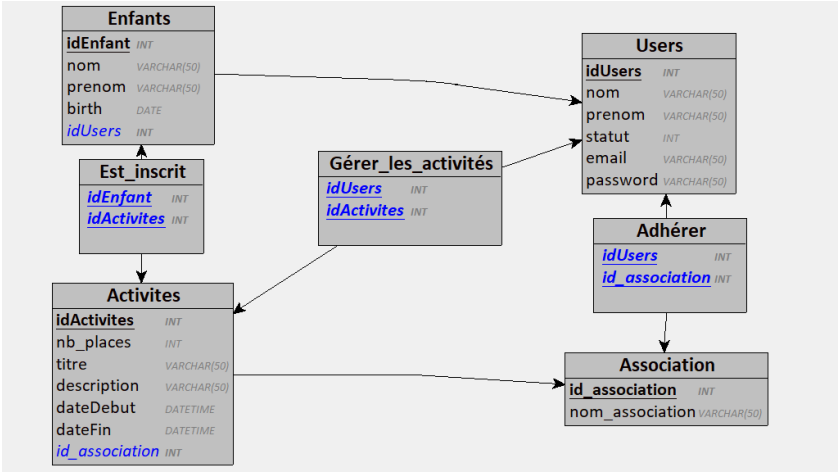
Le but de ce projet est donc de réaliser un site web, dans lequel il y a la possibilité de se connecter/créer un compte, et ajouter ses enfants pour ensuite les inscrire aux différentes activités créées par les gestionnaires. Pour cela, il faut être adhérent à l'association qui gère cette activité. En cas de problème, nous pouvons contacter un administrateur qui a les droits pour modifier certains attributs de notre compte.

Nous avons nommé notre site « Acti'scol ». Acti fait référence aux activités proposées, et scol est le diminutif de scolaire, car le site doit servir à des parents.

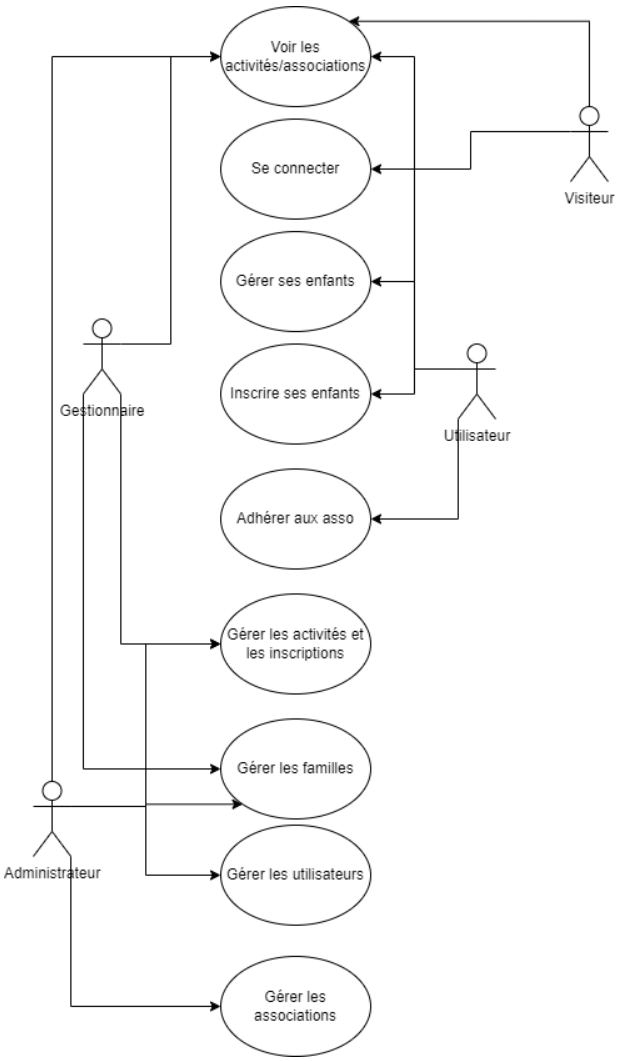
C. Configuration utilisée

Tous les schémas sont présents dans le sous-dossier « Schémas », du dossier Compte rendu.

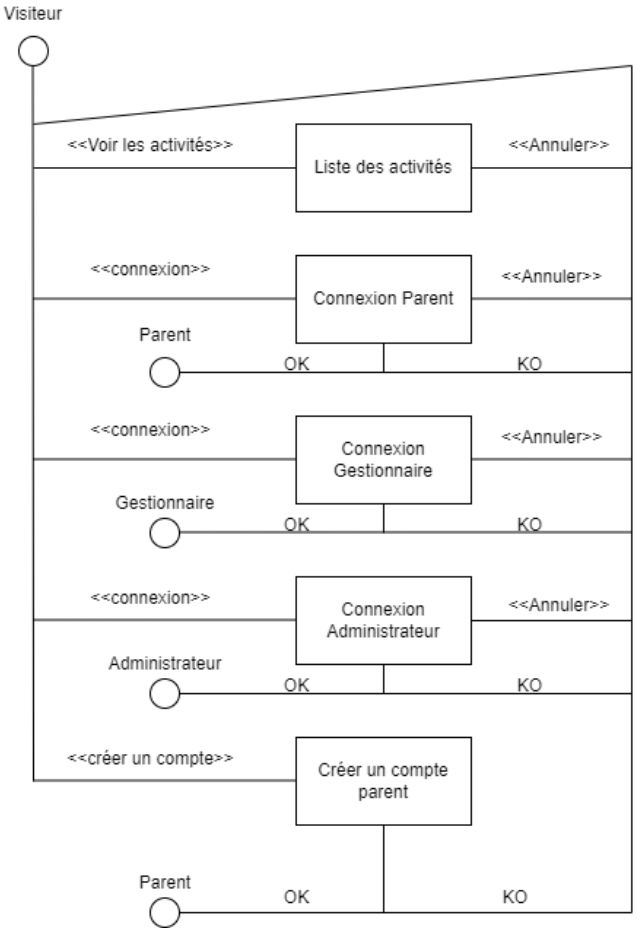
Voici le MLD du projet :



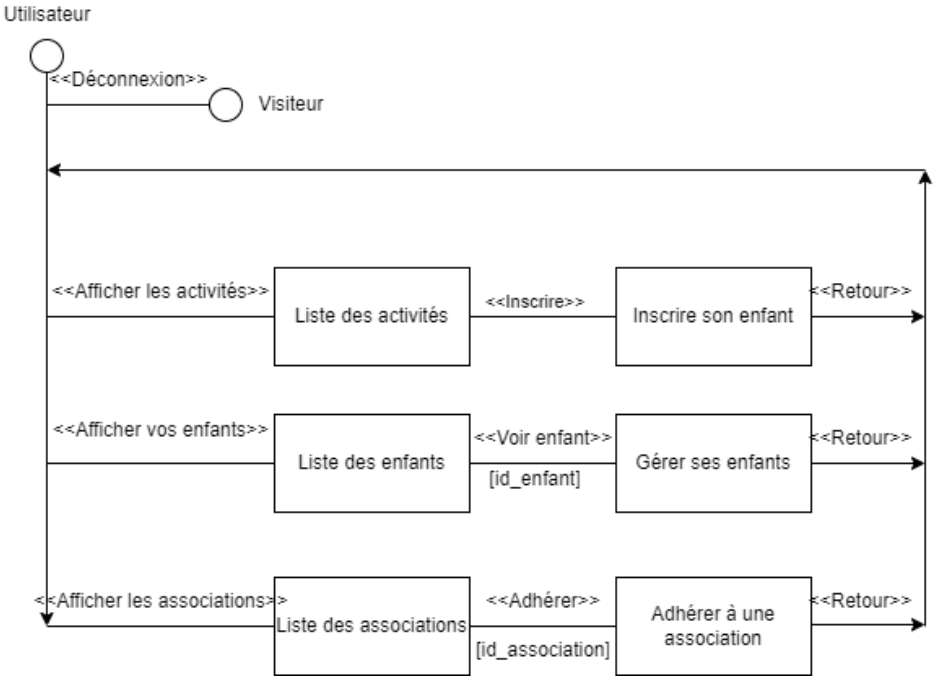
Voici le diagramme des acteurs :



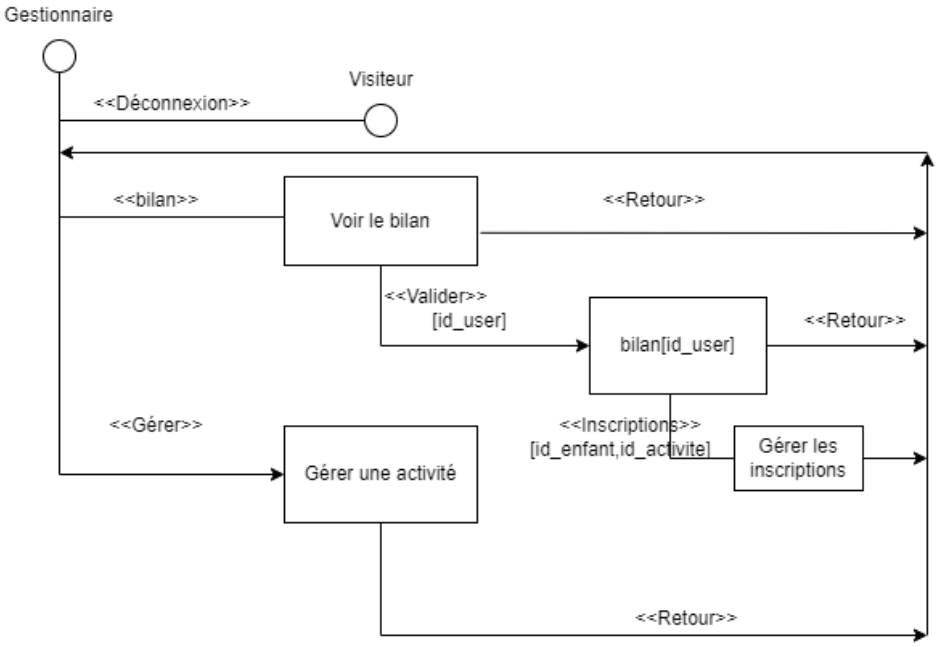
Voici le diagramme d'un visiteur (lorsque l'on n'est pas connecté) :



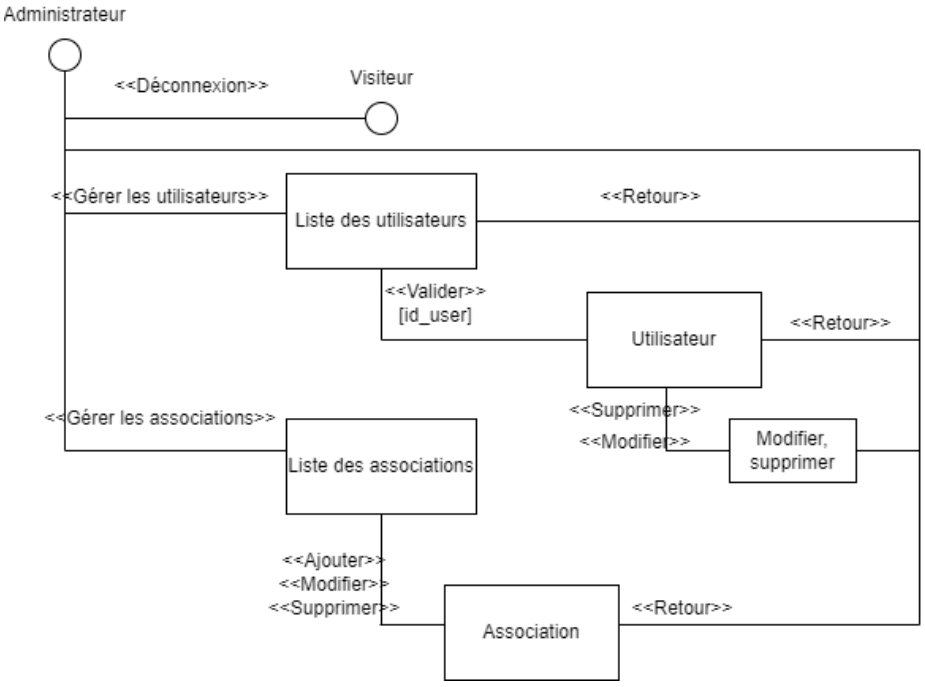
Voici le diagramme d'un utilisateur connecté (utilisateur normal, statut = 1) :



Voici le diagramme d'un gestionnaire (statut = 2, rajouter à ce diagramme celui de l'utilisateur) :



Voici le diagramme d'un administrateur (statut = 3, rajouter à ce diagramme celui du gestionnaire et de l'utilisateur) :



D. Comment l'utiliser

Vous verrez par la suite les différents types d'utilisateurs présents dans notre site. Pour les tester, voici les comptes à utiliser :

- **Utilisateur :** Pseudonyme = Utilisateur ; Mot de passe = motdepasse
- **Gestionnaire :** Pseudonyme = Gestionnaire ; Mot de passe = motdepasse
- **Administrateur :** Pseudonyme = Administrateur ; Mot de passe = motdepasse

Lien de la VM : http://10.5.2.25/~duva0041/projet0303/projet-info303/projet/public/

Lien du projet sur GIT : theo0008 / Projet Info0303 · GitLab (univ-reims.fr)

II. Les parties

A. Non connecté

Lorsque nous arrivons sur le site (et donc que nous ne sommes pas connectés), nous avons quand même plusieurs options disponibles.

Premièrement : La possibilité de voir les activités, et d'en afficher une en détail sur cette page, et rien d'autre.

Deuxièmement : Voir les différentes associations présentes dans la commune

Troisièmement : Nous connecter via un formulaire Jetstream, grâce à notre pseudonyme et à notre mot de passe.

Quatrièmement : Créer un compte via un formulaire Jetstream. Pour cela il faut renseigner plusieurs informations : Un pseudonyme (unique par utilisateur), un prénom, un nom, une adresse mail (unique par utilisateur), et enfin un mot de passe (à confirmer).

Cinquièmement : Afficher les informations du site de la commune (explications, créateurs, logos…).

B. Utilisateur

Lorsque nous sommes connectés, mais que nous sommes un utilisateur normal (juste un parent, donc le statut est égal à 1 dans la base de données), nous avons de nouvelles options disponibles, et les boutons de connexion/création de compte ne sont plus affichés, mais un bouton de déconnexion les remplace.

Premièrement : Dans la liste des activités, un nouveau bouton apparait : celui pour inscrire un enfant. Il faut donc renseigner l'enfant souhaité dans le formulaire, puis valider.

Deuxièmement : Adhérer/désadhérer aux associations, via un bouton dans la liste, ou dans l'affichage d'une association. Ces boutons appellent un script, qui permet d'enregistrer/supprimer notre adhésion dans la table pivot.

Troisièmement : L'affichage de notre compte. En effet, sur cette page toutes nos informations sont mentionnées (sauf le mot de passe bien évidemment), et également la possibilité d'afficher nos enfants, de les modifier, supprimer, et d'en ajouter un dans la liste grâce à un bouton.

C. Gestionnaire

Si nous sommes un gestionnaire, c'est-à-dire que notre statut est 2 dans la base de données, alors nous aurons la possibilité d'éditer, de supprimer, et d'ajouter une nouvelle activité.

Nous avons fait le choix de ne pas lier l'activité à un gestionnaire, car le gestionnaire n'est pas présent physiquement lors de l'activité, c'est juste un employé de la mairie qui enregistre l'activité, donc si le gestionnaire change/disparaît il ne faut pas que l'activité disparaisse également, car elle a bien eu (ou aura) lieu.

De plus, un gestionnaire a accès à la liste des utilisateurs/des enfants, pour ensuite voir le bilan (inscriptions de chaque enfant), qu'il peut d'ailleurs imprimer.

D. Administrateur

Pour le dernier type d'utilisateur, de statut 3, c'est-à-dire un administrateur, il y a plus d'options que les autres. Dans la liste des utilisateurs, il peut les modifier/supprimer (mais ne peut pas voir leur mot de passe). De même pour les enfants, nous pouvons en créer un et le lier à un parent, puis supprimer ses inscriptions si besoin. Un administrateur peut également inscrire n'importe quel enfant à une activité, si cela est possible (si son parent est adhérent, s'il reste de la place, et si les inscriptions ne sont pas fermées, et s'il n'est pas déjà inscrit).

Un administrateur peut donc avoir accès à tout ce qui concerne la base de données, sauf les mots de passe, c'est lui qui gère les gestionnaires, ça pourrait donc être le cas d'un maire par exemple.

III. Conception

A. Les listes

Dans toutes les listes (utilisateurs, enfants, activités et associations), l'utilisateur peut sélectionner certains types, par exemple dans les activités, seulement les prochaines disponibles, car on ne peut s'inscrire à une activité que si elle a lieu dans plus d'une semaine. Ou alors, s'il est connecté, il peut n'afficher que les activités auxquelles il adhère, voir même les activités auxquelles il adhère et qui sont encore disponibles (s'il reste encore au moins une place et que les inscriptions ne sont pas fermées). Pour les gestionnaires/administrateurs, dans la liste des utilisateurs, ils peuvent ne sélectionner que les utilisateurs normaux et dans la liste des enfants, ils peuvent sélectionner en fonction d'un parent. La pagination, la recherche et le tri sont présents.

B. Fonctions particulières

Dans les contrôleurs, nous avons rajouté des fonctions. Par exemple, la fonction « adhérant » dans le contrôleur des associations. Elle permet d'afficher les adhérents de l'association souhaitée (pour les gestionnaires/administrateurs). Il a donc fallu rajouter cette méthode dans le fichier web.php. Il y a la même chose avec une liste des enfants accessible dans le fichier .show d'un utilisateur, puis la liste des inscriptions dans le fichier .show d'un enfant. Cela permet donc de générer le bilan de chaque famille et de l'imprimer.

C. Sécurité

Toutes les sécurités nécessaires sont présentes, dans les contrôleurs ou dans les vues. Par exemple, un utilisateur normal ne peut pas modifier une activité, un visiteur ne peut pas accéder à la page Compte, un gestionnaire ne peut pas supprimer d'utilisateur, etc.

Pour ce faire, nous regardons l'attribut Statut de l'utilisateur : Auth::user()->statut, puis nous ajoutons une condition. De plus, dans la liste de nos enfants, il est impossible d'afficher un enfant qui n'est pas à nous, même en changeant l'url.

Concernant la création (d'activité ou d'enfant), il y a également des sécurités (des vérifications) grâce aux StoreRequests. Par exemple, la date de naissance des enfants ne peut pas être postérieure à la date actuelle, et inversement, une activité ne peut pas avoir une date de début inférieure à la date de la semaine prochaine, car personne ne pourrait s'inscrire. De plus, la date de fin ne peut pas avoir lieu après la date de début. Tout cela est également présent dans les fichiers .edit, avec des messages d'erreurs personnalisés.

D. Base de données

Nous avons créé des migrations, pour créer les tables (enfants, activités, associations), modifier la table users (rajouter un nom, un prénom, un identifiant), et créer des tables pivot (pour adhérer à une association, et inscrire un enfant à une activité). Tout cela est relié aux différents modèles/contrôleurs prévus.

Cependant, avec les tables pivot nous avons rencontré un problème : le contrôleur correspondait à une table avec les deux éléments au singulier, cependant Laravel cherchait dans une table avec le deuxième élément au pluriel. Nous avons donc dû renommer la table comme souhaité avant les instructions, puis à la fin un deuxième renommage pour revenir avec l'ancien nom.

E. Les dates

Comme dit précédemment, les dates sont importantes dans les conditions, pour éviter d'avoir un enfant qui n'est pas encore né, etc. Dans le Seeder de l'activité, nous utilisons plusieurs fonctions strtotime pour créer une date de début postérieure à la date actuelle, puis pour faire rajouter plusieurs heures à cette date et obtenir la date de fin de l'activité.

De plus, lors de l'affichage nous utilisons plusieurs formats pour l'affichage des dates. Par exemple dans la liste, pour obtenir l'affichage « 3 décembre 2022 à 8h35 », le formatObject est le suivant : 'd MMMM Y' à 'H'h'm', mais dans le fichier .show, nous avons en plus de cela le jour en toutes lettres (samedi), pour cela nous avons rajouté 'EEEE' au début.

F. Peuplement

Pour le peuplement de la base de données, voici l'ordre que nous utilisons : Les utilisateurs (3 prédéfinis et 35 grâce au factory), puis leurs enfants (100), car ils ont besoin d'un user_id, donc il faut que les utilisateurs soient créés avant. Ensuite les 5 associations sont créées, puis grâce à une boucle et des conditions, des utilisateurs deviennent adhérents (au hasard). Après cela, les activités sont créées (donc en dernier), puisqu'elles ont besoin d'une association_id. Cela permet alors de générer aléatoirement des inscriptions avec des enfants (s'il reste encore des places et que le parent est adhérent).

G. Remarques

Dans la dernière partie de cette rubrique, nous allons expliquer ce que nous pourrions rajouter dans notre site. Premièrement un âge minimum/maximum pour chaque activité. Et dans le même esprit, un âge maximum pour un enfant, car la date de naissance possède uniquement une condition pour ne pas être supérieure à la date actuelle, un enfant peut très bien avoir 80ans.

De plus, il n'y a pas de paramètres dans le compte, c'est-à-dire que si l'utilisateur veut changer son adresse mail ou son identifiant, seul un administrateur peut le faire, et il est impossible de changer son mot de passe.

IV. Commandes

Pour installer le projet sur la VM, il faut utiliser plusieurs commandes.

- **composer install** -> créer les bibliothèques .json.
- **copy .env.example .env** -> créer le fichier .env, puis le modifier pour se connecter à la base de données.
- **php artisan key:generate** -> générer la clé unique pour accéder au dossier storage.
- **php artisan storage:link** -> relier la clé à Laravel pour ensuite utiliser des fichier (situés dans storage) dans les vues.
- **php artisan migrate --seed** -> créer les tables puis les remplir grâce aux seeders précisés dans DatabaseSeeder. Avec :fresh cela permet de supprimer les tables avant de faire le peuplement.
- **npm install** -> installer Jetstream.
- **npm run build** -> relier Jetstream à notre projet Laravel.

V. Design

Concernant le design du site, nous avons gardé le même format que celui présent dans les TP. Cependant, un thème sombre est présent (également dans les pages Jetstream). De plus, nous avons rajouté une icone au site, que ce soit dans l'onglet, ou dans le titre de la page (en le cliquant on revient à l'accueil).

Sur certaines pages (les .show) est présent un bouton « page précédente », car par exemple, si un gestionnaire va dans la liste des utilisateurs, puis en affiche un, puis appuie sur le bouton pour voir ses enfants, puis appuie sur le bouton pour voir les inscriptions de cet enfant, en utilisant le bouton retour il va arriver dans la liste des enfants, alors qu'à la base il ne vient pas de cette page. Pour ce faire, le bouton appelle un script embarqué javascript, qui permet d'aller à la page précédente dans l'historique.

Un arrière-plan est présent, et le logo de Jetstream a été changé par une autre image svg, et les formulaires d'authentifications ont été modifiés pour correspondre à la base de données.

VI. Conclusion

Nous trouvons que notre site web est bien réussi, et répond au sujet. Au début, ce projet paraissait compliqué, mais finalement en reprenant les TP c'était très accessible et Laravel rend les requêtes plus faciles. L'utilisation d'un git et la mise en place d'un cahier des charges permettent de travailler efficacement à plusieurs.

Voici le logo de notre site Acti'scol :

Antoine Duval & Antoine Théologien