R4.01 - Programmation Web with PHP
Projet - Implantation d'une application MVC avec Laravel
Application de gestion hospitalière

Ricardo Uribe Lobello

25 février 2025

Application à implanter

Dans ce projet, il s'agit d'implanter une application web avec architecture monolithique Modèle-Vue-Contrôleur (plutôt MVVM) avec le framework Laravel. Pour le faire, il faudra utiliser le diagramme de classes suivant :

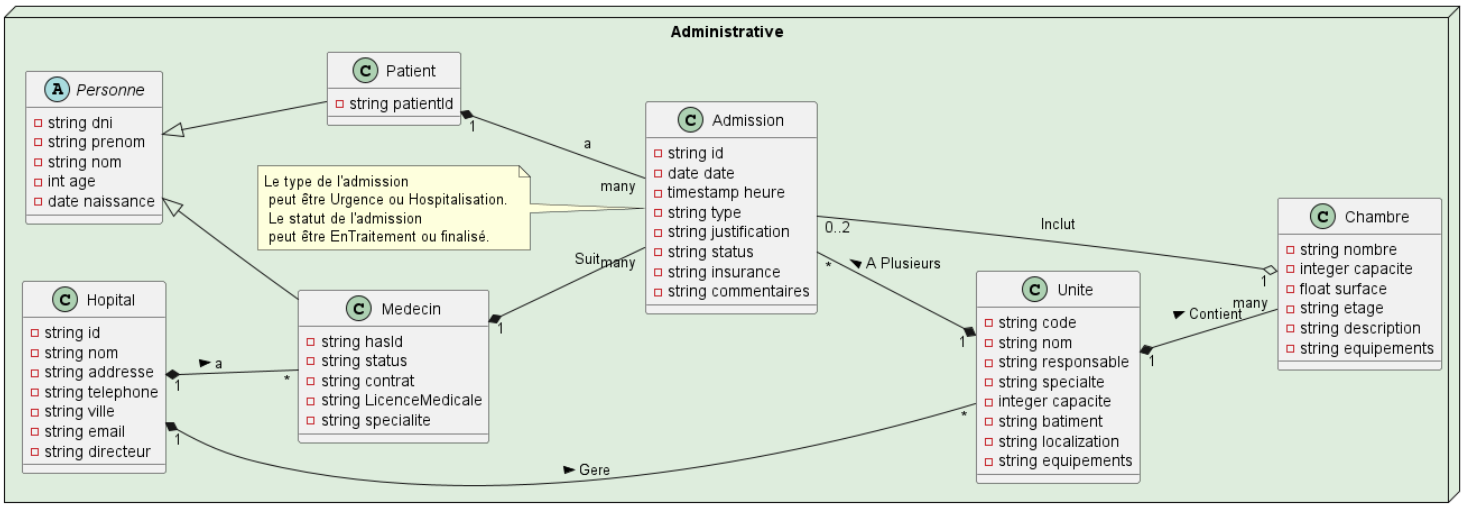

Figure 1: Diagramme UML de l'application

Fonctionnalités à implémenter

L'application doit fournir les fonctionnalités de base suivants:

- **Opérations CRUD de base:** Il faudra fournir les pages et formulaires nécessaires pour impléménter les opérations de création, lecture, mise à jour et élimination (soft delete) sur toutes les classes suivantes:
  - Hôpital;
  - Medecin;
  - Unite;
  - Chambre;
  - Patient;

- Admission.

Il faut considérer toutes les associations entre les différents concepts. Par exemple, un medecin doit être associé à un hôpital, une admission à une unité, à un medecin et à un patient, etc. Vous pouvez travailler dans l'ordre qui vous est proposé ci-haut. Ces associations seront représentées via des clés étrangères dans la base de données mais aussi dans le modèle ORM Eloquent.
Il est clair qu'il faudra créer tous les routes, formulaires et pages d'affichage nécessaires pour pouvoir manipuler les informations présentées ci-haut. Par exemple, pour l'hôpital:

- **http://localhost/hopital (méthode GET pour afficher le formulaire et méthode POST pour le soumettre):** routes pour créer un hôpital;
- **http://localhost/hopital/id (méthode GET):** route pour consulter l'hôpital avec l'identifiant fourni;
- **http://localhost/hopital/id (méthode DELETE):** route pour éliminer l'hôpital avec l'identifiant fourni;
- **http://localhost/hopitaux (méthode GET):** route pour consulter la liste des hôpitaux;

- **Navigation:** Une interface permettant de naviger sur les différentes pages. Il peut s'agir d'un entête contenant le menu de navigation.
- **Fonctionnalité Ajax:** Dans la page de consultation d'une unité, il faudra utiliser Ajax pour pouvoir charger de manière dynamique et à la demande, les informations concernant les chambres associées à l'unité et les admissions qui lui ont été attribuées. Si vous voulez, à la place d'Ajax, vous pouvez utiliser la fonctionnalité **livewire de Laravel.**
- **Securité:** Un module d'enregistrement, autentification et identification des utilisateurs (voir la documentation sur Laravel Breeze). Il faudra que la session affiche le nom de l'utilisateur courant a coté du ménu de navigation.
IMPORTANT: Pour le moment, nous allons travailler comme si tous les utilisateurs sont des administrateurs.
- **Bonus pour les binômes ou à faire obligatoirement pour les groupes de 3 étudiants**:
  - **Suivi des opérations:** Il faudra ajouter l'identifiant de chaque utilisateur dans les enregistrements qui ont été créés, modifiés, ou soft deleted par l'utilisateur. Cela permerttra de garder une trace de qui a modifié quoi.
  - **Log:** Il faudra ajouter un log d'opérations qui ont été réalisées, chaque opération réalisée doit être enregistrée dans un log afin de pouvoir suivre sa trace. Regarder l'interface Log de Laravel.
  - **Rôles de l'application:** Il faudra ajouter au moins un autre type d'utilisateur qui n'aura pas d'accès à toutes les fonctionnalités de l'application. Par exemple, un patient n'aura pas accès qu'à ses informations, la liste et les détails des informations de ses admissions. Il ne pourra pas faire des modifications. Vous pouvez consulter les modèles de rôles inclus avec Laravel.
  - **Utilisation de livewire ou d'Ajax:** Vous devrez proposer une autre fonctionnalité (au délà de celle demandée) qui permette d'utiliser d'Ajax dans l'application afin d'améliorer l'interactivité d'une page de l'application.

Modalité de travail

Groupes: Vous pouvez travailler tout seul, en binômes et, exceptionnellement à trois. Il faudra profiter des séances de TP et de TD d'ici à la fin du cours pour avancer le plus possible sur ce projet.

Gestion du code source: Vous devez travailler avec un dêpot GIT qui vous sera fourni par votre intervenant.

Base de données: Vous pouvez utiliser une base de données locale mais si cela n'est pas possible, vous pouvez toujours vous connecter à la base de données sur **Pedaweb**. Suivez les instructions de votre intervenant pour le faire.

Avancement: Il faudra essayer d'avancer dans chaque séance de TP. Une absence d'avancement dans les séances TP peut se voir reflété dans la note finale, même si vous implémentez tous ce qui est demandé..

Rendu: Dans la dernière séance du TP du 26 mars, vous devrez me faire une présentation de votre application et du code que vous avez écris dans environ 20 minutes. Pendant la soutenance, je pourrais vous poser des questions individuelles afin de mesurer votre comprehension du code qui a été réalisé.
Ensuite, vous devrez créer un tag git pour la version finale de votre projet afin que je puisse le consulter ensuite pour arriver à une note finale de projet.

Informations techniques pour commencer votre projet Laravel

Les informations techniques pour commencer votre projet se trouvent ici sur le cours en AMETICE.

Tâches à faire

1. **Configuration initial et migrations:** Nous allons commencer par créer le projet et la configuration initiale du projet et les migrations correspondantes pour toutes les tables du projet.

   (a) **Important: A cause des configurations des machines virtuelles de l'IUT, il faudra que faire attenttion à conserver vous récupérez votre code à la fin de chaque séance de cours. De plus, vous devez aussi exporter la base de données et la récréer à chaque séance. Les machines virtuelles de l'IUT ne vont pas nécessairement conserver vos fichiers.**

2. Ensuite, pour chaque classe dans le diagramme :

   (a) Il faudra implémenter le modèle de données. Pour cela, il faut créer les classes modèle. Cela nous donnera l'accès aux opérations de base CRUD (Création, modification et élimination). Pour chaque classe, il faudra créer une table dans la base de données contenant :

      i. Tous les champs énoncés pour la classe.
      ii. Les clés étrangères issues des associations de chaque classe.
      iii. Pour chaque table, il y a trois champs additionnels : **created_at**, **updated_at** et **deleted_at** du type **Timestamp** pour faire le suivi des modifications des enregistrements et pouvoir implanter des **soft deletes**. Les enregistrements éliminés ne doivent pas être visibles que pour l'administrateur de l'application.
      Attention: ces trois champs doivent pouvoir être NULL.

3. Ensuite, il faut créer le ou les contrôleurs afin de traiter les requêtes et créer les réponses. Pour cela :

   (a) Il faut créer les routes pour chaque méthode du contrôleur.

   (b) Il faut créer les méthodes dans le contrôleur.

   (c) Il faut créer les pages de l'application avec Blade.

   (d) Vous pouvez commencer par retourner du texte ou du json pour tester que tout se passe bien.

4. Il faudra aussi créer les réponses pour chaque requête dans l'application. Il faudra considérer que, dans chaque réponse :

   (a) Il faut avoir un entête permettant de naviguer dans les différentes options de l'application. L'entête doit avoir une option pour fermer la session et le nom et prénom de l'utilisateur doit toujours être affiché.

   (b) Il serait bien d'avoir également un pied de page qui doit nous permettre de revenir vers le haut de la page.

   (c) Il faudra avoir une partie centrale contenant les informations de la base de données.

   (d) **Les couleurs et style de l'application reste à votre discrétion mais attention à garder le respect dans ce que vous proposez.**

5. L'objet session de l'application doit, à tout moment, conserver les informations de base d'un utilisateur, en particulier le nom d'utilisateur et surtout le rôle qui sera utilisé dans la partie sécurisation de l'application.

   (a) Vous pouvez connecter les utilisateurs avec les medecins ou patients mais cela n'est pas obligatoire que si vous devez implémenter la partie bonus.