IUT Nancy-Charlemagne – BUT Informatique S5 – DWM

Développement Web Serveur avancé

TD2 : Programmation du composant rendez-vous

Dans ce TD, on s'intéresse à la programmation du composant (ou service métier) de gestion de prise de rendez-vous. Il s'agit du composant le plus important, celui qui permet de créer, valider et annuler/modifier les rendez-vous.

Il utilise les composants Praticien et Patient, afin de connaître d'une part la personne qui prend le rendez-vous, et les contraintes du praticien d'autre part.

Contraintes et recommandations

Le composant Rendez-vous fait partie de la couche Métier. Les namespaces et l'organisation de votre projet doivent refléter cette appartenance. Un composant (ou service) est une classe qui implante une interface : il faut donc créer en PHP l'interface et la classe.

Le composant Rendez-vous doit utiliser le composant Praticien pour connaître les contraintes du praticien, sa spécialité, et le lieu où il exerce. Ce composant doit être passé en paramètre du constructeur du composant Rendez-vous. Dans une première version, on ne gère pas les patients, et on suppose que le patient est connu et que l'on dispose de son identifiant.

Les deux composants devront, à terme, utiliser des bases de données différentes : les praticiens seront stockés dans une base et les rendez-vous seront stockés dans une autre. Dans un premier temps, on n'utilise pas de base de données, mais on simule leur utilisation en stockant les données dans des tableaux.

Tests : il est nécessaire de tester les différentes méthodes du composant pour s'assurer qu'elles fonctionnent correctement. Les tests doivent être programmés dans le répertoire `./tests`. Ces tests peuvent être réalisés à l'aide de PHPUnit ou de scripts.

Exercice 1: consulter un rendez-vous

Programmer la méthode qui permet d'obtenir les informations associées à un rendez-vous. Elle reçoit en paramètre l'identifiant du rendez-vous et retourne un DTO. Elle utilise un Repository pour accéder aux données, et interroge le composant Praticien pour obtenir les informations sur le praticien associé au rendez-vous.

Elle déclenche une exception si le rendez-vous n'est pas trouvé.

Exercice 2 : créer un rendez-vous

Programmer la méthode `creerRendezvous()` qui permet de créer un rendez-vous. Elle reçoit en paramètres toutes les données nécessaires à la création du rendez-vous.

Il faut bien sûr vérifier que le rendez-vous est possible :

- le praticien existe,
- la spécialité indiquée pour le rendez-vous fait bien partie de la liste des spécialités du praticien,
- le praticien est disponible à la date et à l'heure demandées.

Si le rendez-vous est impossible, une exception est déclenchée.

Une fois le rendez-vous créé et sauvegardé dans le repository, un objet DTO est retourné avec toutes les informations disponibles.

Exercice 3 : annuler un rendez-vous

Programmer une méthode pour annuler un rendez-vous. Le rendez-vous est désigné par son ID. On souhaite cependant conserver la trace des rendez-vous qui ont été annulés. L'annulation consiste donc à marquer le rendez-vous comme annulé.

Exercice 4 : lister les disponibilités du praticien

Programmer une méthode qui retourne la liste des disponibilités d'un praticien sur une période donnée. Dans un premier temps, on fait l'hypothèse que tous les praticiens partagent les mêmes contraintes sur les rendez-vous qu'ils acceptent : mêmes jours de consultation, mêmes horaires, mêmes durées de rendez-vous.

Ces données doivent cependant être délarées dans des constantes pour pouvoir être modifiées facilement. La méthode retourne un tableau de créneaux horaires disponibles, chaque créneau étant un objet `DateTime`.

Exercice 5 : modifier un rendez-vous

Programmer une ou plusieurs méthodes pour modifier un rendez-vous. On considère que l'on ne peut modifier que la spécialité ou le patient du rendez-vous. Les autres types de modifications sont réalisées en annulant le rendez-vous et en en créant un nouveau.

Par ailleurs, ajouter une fonctionnalité de logging dans le composant de prise de rendez-vous, afin de conserver une trace de toutes les opérations réalisées sur les rendez-vous (modification, annulation…) Pour cela, vous devez utiliser un logger conforme à PSR-3 (par exemple Monolog). Il faut donc l'injecter dans le composant au travers du constructeur.

Exercice 6 : gérer le cycle de vie des rendez-vous

Un rendez-vous est créé initialement avec un statut "prévu". Il peut ensuite être annulé par le patient ou par le praticien. Lorsque la consultation est effective, le rendez-vous est marqué comme "honoré". Il peut ensuite être payé, et transmis aux organismes sociaux. Si le patient n'honore pas le rendez-vous, le statut est "non honoré".

Programmer les méthodes permettant de réaliser ces transitions.