IUT Nancy-Charlemagne – BUT Informatique S5 – DWM

Développement Web Serveur avancé

TD1.3 - Partie 1 : middlewares, DTO et validation de données

L'objectif de cette première partie du TD 1.3 est de comprendre et utiliser des middleware dans une application Slim, et d'appliquer le principe des middleware pour valider des données et créer un DTO d'entrée dans le cas d'une requête de création d'un rendez-vous dans le projet Toubilib.

Exercice 1 : DTO, interface et service

L'objectif est de programmer le service gérant les rendez-vous pour ajouter le cas de la création d'un nouveau rendez-vous. Les données transmises par le créateur du RDV sont les suivantes : - l'ID du praticien, - l'ID du patient, - la date et l'heure du rendez-vous, - le motif de visite, - la durée du rendez-vous, en minutes.

Vous devez prévoir : - créer un DTO `InputRendezVousDTO` permettant de tranmettre les données d'un rendez-vous au service, - adapter l'interface `ServiceRendezVousInterface` pour y ajouter la méthode `creerRendezVous(InputRendezVousDTO $dto)`, - adapter le service `ServiceRendezVous` pour y ajouter la méthode `creerRendezVous(InputRendezVousDTO $dto)`, qui implémente le cas d'utilisation de création d'un rendez-vous, - adapter le l'interface et le repository `RendezVousRepository` pour y ajouter la méthode de sauvegarde d'un rendez-vous dans la base,

Dans un premier temps, faites la création/sauvegarde du rendez-vous dans la base de données, sans validation des données.

Exercice 2 : validation métier des données

Compléter le service `ServiceRendezVous` pour y ajouter la validation métier des données du rendez-vous, en vérifiant que : - le praticien et le patient existent, - le motif de visite fait partie des motifs pour ce praticien, - le créneau horaire demandé est valide : jour ouvré et horaire possible (on fera l'hypothèse que tous les praticiens travaillent du lundi au vendredi, de 8h à 19h), - le praticien est disponible pour le créneau horaire demandé,

Exercice 3 : middleware de création du DTO / validation des données

L'objectif est de créer un middleware qui va valider les données d'entrée de la requête et créer le DTO `InputRendezVousDTO` à partir des données de la requête.

- identifier les contrôles et validation des données reçues dans une requête HTTP nécessaire pour pouvoir créer le DTO : présence, type, format, protection des données.
- créer un middleware dont le rôle sera de récupérer les données de la requête, de les valider, et

de créer le DTO `InputRendezVousDTO` à partir des données valides. Dans quelle partie de l'architecture hexagonale se situe ce middleware ? Comment transmettre le DTO créé ?

Exercice 4 : Action de création d'un rendez-vous

Programmer maintenant l'action de création d'un rendez-vous ; elle récupère le DTO créé par le middleware, appelle le service `ServiceRendezVous` pour créer le rendez-vous, et retourne une réponse HTTP adéquate. Elle doit détecter les cas d'erreur et retourner un statut de retour adéquat, ainsi qu'un message d'erreur au format JSON.

Rappels : - appliquer les principes RESTful, notamment en utilisant la méthode HTTP et l'URL adéquates, et en renvoyant un statut de retour correct, - utiliser le conteneur d'injection de dépendances pour créer les différents objets (action, middleware, service, repository, connexion PDO…)

TD1.3 - Partie 2 : gestion métier des rendez-vous

L'objectif de cette seconde partie du TD 1.3 est de compléter les fonctionnalités du projet toubilib concernant les rendez-vous.

Exercice 1 : annuler un rendez-vous

L'objectif est de compléter le service gérant les rendez-vous pour ajouter le cas de l'annulation d'un rendez-vous. On procédera comme suit :

1. programmer une méthode d'annulation d'un rendez-vous sur l'entité `RendezVous` (par exemple, `annuler()`), qui met à jour l'état du rendez-vous ; la méthode vérifie que peut être annulé (par exemple, que le rendez-vous n'est pas déjà annulé, et que la date du rendez-vous est dans le futur). Si ce n'est pas le cas, la méthode lève une exception métier.
2. adapter l'interface `ServiceRendezVousInterface` pour y ajouter la méthode `annulerRendezVous(int $idRdv)`, qui prend en paramètre l'identifiant du rendez-vous à annuler,
3. adapter le service `ServiceRendezVous` pour y ajouter la méthode `annulerRendezVous(int $idRdv)`, qui implémente le cas d'utilisation d'annulation d'un rendez-vous ; cette méthode charge le rendez-vous en utilisant le repository, appelle la méthode d'annulation de l'entité `RendezVous`, puis sauvegarde la modification en faisant appel au repository. Si le rendez-vous n'existe pas, la méthode lève une exception métier.

Programmez ensuite une action de l'API qui permet d'annuler un rendez-vous. Cette action appelle la méthode d'annulation du service `ServiceRendezVous`, et retourne une réponse HTTP adéquate. Elle doit détecter les cas d'erreur et retourner un statut de retour adéquat, ainsi qu'un message d'erreur au format JSON. Prévoir enfin la route correspondante, qui doit se conformer aux principes RESTful (méthode HTTP, URL, statut de retour).

Exercice 2 : agenda d’un praticien

On veut programmer la fonctionnalité permettant à un praticien de consulter son agenda, c’est-à-dire la liste de ses rendez-vous sur une période donnée. Par défaut, on retourne les rendez-vous de la journée en cours, mais on doit pouvoir demander une période quelconque. On retourne la liste des rendez-vous du praticien, avec ses informations complètes - date, heure, durée, motif de visite, état du rendez-vous (annulé ou non), accompagné d’un lien vers les données du patient concerné. Vous devez adapter l’ensemble des composants nécessaires pour réaliser cette fonctionnalité : repository pour les rendez-vous, service de gestion des rendez-vous, action de l’API, route correspondante.