IUT Nancy-Charlemagne – BUT Informatique S5 – DWM

Développement Web Serveur avancé

TD1.1 : Architecture hexagonale

L'objectif du TD est de comprendre les principes de l'architecture hexagonale et de l'inversion de dépendances, et de mettre en oeuvre ces principes dans le contexte du projet Toubilib.

Exercice 1 : Analyse de l'architecture de l'application JIRA

Récupérez le code de l'application JIRA : https://gitlab.univ-lorraine.fr/perrin15/jira.git.

Analysez l'architecture de l'application JIRA, en identifiant les composants dans chacune des trois couches et les dépendances entre ces composants. Vous devez produire un diagramme de classes complet, ainsi qu'une description textuelle de l'architecture de l'application. Expliquez comment est réalisée l'inversion de dépendances entre l'infrastructure de persistance et le domaine métier, ainsi que la manière dont les composants sont injectés dans les actions du contrôleur.

Exercice 2 : Projet Toubilib, opérations préliminaires

Analysez le sujet du projet Toubilib disponible dans Arche et récupérez le squelette, contenant notamment la structure et les données de départ des bases de données, ainsi qu'un fichier `docker-compose.yml`.

1. identifiez les différentes bases de données utilisées dans le projet, et faites un modèle du domaine UML pour chacune de ces bases de données,
2. adaptez le fichier `docker-compose.yml` à votre environnement, démarrez les services Docker, puis créez les bases de données,
3. identifiez les namespaces des différents composants du projet,
4. installez les dépendances nécessaires au projet à l'aide de composer.

Exercice 3 : Projet Toubilib, fonctionnalité 1

L'objectif de l'exercice est de programmer la version initiale de la fonctionnalité 1 du projet Toubilib, qui consiste à lister les praticiens, en mettant en place scrupuleusement les principes de l'architecture hexagonale et de l'inversion de dépendances. On traite pour l'instant uniquement la liste complète des praticiens, sans filtrage ni pagination. De plus, chaque praticien est restitué avec ses informations de base (nom, prénom, ville, email, spécialité). Les motifs de visites, moyens de paiement et structure d'appartenance ne sont pas encore gérés.

Vous devez prévoir :

1. l'entité métier `Praticien`,

2. le service métier `ServicePraticien` qui implante l'interface `ServicePraticienInterface` et réalise un seul cas d'utilisation au travers de la méthode `listerPraticiens()`,
3. un DTO `PraticienDTO` permettant au service de retourner les informations de base d'un praticien,
4. un adaptateur de persistance `PraticienRepository` qui implante l'interface `PraticienRepositoryInterface` et réalise la récupération des praticiens depuis la base de données.

Testez ce service dans un script PHP, puis utilisez ce service dans une action de l'application Slim. Déclarez la route correspondante dans le fichier de configuration des routes de l'application, et faites en sorte que l'action renvoie la liste des praticiens au format JSON. Veillez à respecter les principes RESTful, notamment en utilisant la méthode HTTP et l'URL adéquates, et en renvoyant un statut de retour correct.