**TP Structure Application**

**Nous allons repartir de notre API d’étudiants**

**1/ Repository encore!**

Vérifiez que tous vos repository sont dans un package dédié. Les implémentations doivent également s’y trouver, idéalement dans un sous package nommé “impl”

**2/ Ajout des Services**

Déplacer tout le code qui ne concerne pas la persistance, ou la gestion des requêtes/réponses dans des classes de type Service.

Ces classes sont dites “de métier”. Aucune information sur le type du stockage ou sur la manière d’exposer les données ne doit y figurer. Ceci permet en principe un changement de l’un ou l’autre très facilement.

Comme pour les Repository, ces classes doivent être découpées en interface + implémentation et stockées dans des packages dédiées.

Dans notre cas des professeurs, les service ne sera qu’un “passe-plat”. Il ne fait qu’appeler le stockage, le repository.

Pour lui donner un sens, nous allons ajouter un comportement à notre API :

La liste des professeurs retournée doit être triée alphabétiquement par nom.

@Service pour définir un Service, à placer sur l’implémentation. Injectez y les éventuels @Repository.

Enfin, injectez les Services dans vos @RestController.

**4/Documentation**

Ajouter de la documentation peut servir à tous les futurs développeurs qui arriveraient sur le projet. Elle pourrait les aider à comprendre l’utilité des différentes classes, méthodes et autres présents dans le code.

Deux types de commentaires en Java :

*// Le commentaire ligne à ligne*

Il sert à donner des indications sur un endroit précis du code, au sein d’une méthode généralement

*/\**

*Le commentaire par bloc*

*\*/*

Il est exportable pour fournir une documentation d’application. Il s’applique au dessus des méthodes et des classes.

*/\*\**

*\* Classe qui sert a ...*

*\*/*

**public class** MaClass {

}

*/\*\**

*\* Methode servant a ...*

*\** ***@param param*** *le parametre sert a...*

*\** ***@return*** *Une liste de chaines representant ...*

*\*/*

List<String> maMethode(String param);

**NB:** Votre IDE ajoute automatiquement des éléments dans les commentaires de méthode pour définir les paramètres et les retours des méthodes.

**NB2 :** Quitte a faire des fautes, vous pouvez éviter les accents dans les commentaires. D’un poste à l’autre (Unix/Windows, etc), ils peuvent créer des problèmes de compilation.

Ajouter des commentaires à votre application, puis générez la documentation Java via la commande :

**mvn javadoc:javadoc**

Cela va utiliser un plugin maven présent par défaut dans votre application.

Résultat obtenu dans <votre\_projet>/target/site/apidocs

**Bonus : DTO != DO**

Vos Controller ne doivent plus retourner des DO (ou entités si vous faites du JPA), mais d’autres objets, à la structure similaire aux existants: CoursDto, MatiereDto,....

Dans ces classes, aucune information sur le système de persistance ne doit apparaître. Ne doivent figurer, au pire, que des informations liées à la sérialisation JSON, ou au validateurs hibernate.

Regroupez ces classes dans un package dédié.

Faites la conversion de l’un à l’autre via au choix :

* Un constructeur
* Une classe “utilitaire”
* Des méthodes statiques
* autre ?