Cave à vin (partie 05)

# TP05 du module 02 – Spring Data

|  |
| --- |
| Ce TP sera un TP fil rouge du cours   * Création d’une hiérarchie d’héritage par jointure |

|  |
| --- |
| **Durée estimée** |
| 1 heure |

Contexte

En vous appuyant sur l’ensemble des concepts abordés dans le cours, mettez en place une application Spring Boot permettant la gestion d’une cave à vin.

L’application de cave à vin sera une application d’API Web sécurisée à la fin des itérations :

* Elle permet de gérer le stock des bouteilles de vins par le propriétaire qui est un administrateur de l’application
* Elle permet de gérer l’achat en ligne des bouteilles par un client
  + Les bouteilles de vin peuvent avoir 3 couleurs : Rouge, Blanc ou Rosé
  + Les bouteilles de vin sont associées aux régions françaises
* Un client et un propriétaire doivent s’authentifier
* Un visiteur est un client qui n’a pas de compte
* Il pourra uniquement consulter le stock

Dans les itérations nous réaliserons uniquement le back de l’application.

Cette application se fera avec Spring Boot.

## Voici le diagramme des cas d’utilisation complet :

Une image contenant texte, diagramme

Description générée automatiquement

## Voici le diagramme des entités à réaliser :

Une image contenant texte, diagramme, ligne, Parallèle

Description générée automatiquement

Voici le diagramme des tables en base de données :

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Énoncé

# Créer et modifier les entités

* Les packages des classes sont fr.eni.cave.bo.client et fr.eni.cave.bo
* Utiliser les annotations de JPA
  + Appliquer les annotations pour la hiérarchie d’héritage par jointure
* Les attributs : pseudo, nom, prenom et password sont remontés au niveau de la classe mère
* Les contraintes sur equals et toString sont aussi remontées sur la classe mère
  + Pour l’entité Client le toString de la classe mère suffit. Tous les attributs tracés sont remontés sur la classe mère
  + Penser à utiliser le paramètre callSuper pour l’annotation @ToString sur Proprio
* Penser à utiliser l’annotation @SuperBuilder sur les 3 entités à la place de @Builder

# Créer les Repository des entités Utilisateur et Proprio

# Créer des tests unitaires

* Une classe de tests unitaires appelée TestHeritage vous est fournie
* Elle contient une méthode initDB annotée @BeforeEach
  + Elle injecte en base 3 utilisateurs :
    - 1 Utilisateur
    - 1 Propio
    - 1 Client
* En vous aidant des tests unitaires des démonstrations et des TP précédents
* Créer 3 tests unitaires :
  + Test\_findAll\_Utilisateur à doit retourner les 3 utilisateurs créés
  + Test\_findAll\_Proprio à doit retourner uniquement le propriétaire créé
  + Test\_findAll\_Client à doit retourner uniquement le client créé
* Voici le genre de traces à obtenir (les retours chariots et décalages sont là pour simplifier la lecture, ils ne font pas parti des traces d’origine)
  + Les utilisateurs :

[ Utilisateur(pseudo=harrisonford@email.fr, nom=Ford, prenom=Harrison),

Proprio(super=Utilisateur(pseudo=georgelucas@email.fr, nom=Lucas, prenom=George), siret=12345678901234),

Utilisateur(pseudo=natalieportman@email.fr, nom=Portman, prenom=Natalie)]

* + Les propriétaires :

[Proprio(super=Utilisateur(pseudo=georgelucas@email.fr, nom=Lucas, prenom=George), siret=12345678901234)]

* + Les clients :

[Utilisateur(pseudo=natalieportman@email.fr, nom=Portman, prenom=Natalie)]