Doctrine TP3 - Relations d'objets et DQL

Dans ce TP nous allons mettre en place des relations entre les différents objets et requêter dessus en DQL

1. Branchez le formulaire d'enregistrement d'un utilisateur
2. Branchez le formulaire de connexion d'un utilisateur
3. À chaque requête nous devons récupérer cet objet utilisateur à partir dans la session afin de lui afficher le contenu qui lui ai autorisé **/!\ Votre utilisateur en session n'est plus traqué par doctrine, vous devez faire un "merge" avec l'entitymanager pour le récupérer**. (faites le dans le bootstrap.php)
4. Créez les relations dans Post et Comment vers "User" dans l'attribut "$author"
5. Créez un lien de Comment vers Post dans l'attribut "$post" et Post vers Comment dans "$comment"
6. Modifiez l'action de sauvegarde d'un Post et d'un Commentaire afin d'y ajouter l'auteur automatiquement à la sauvegarde.
7. Modifiez l'action de sauvegarde d'un commentaire, afin d'y ajouter automatiquement le Post associé.
8. Modifiez l'affichage des commentaires pour n'afficher que ceux du Post passé en paramètre GET (voir TP2)
9. Gérer la suppression de Post/Comments (méthode "remove" de l'entity manager)
10. Créez une entité "PostLike" et "CommentLike" qui permettent d'enregistrer le nombre de likes et dislikes sur un post et un commentaire
    a. PostLike contient : "$post" -> référence vers un Post ; "$user" -> référence vers un user ; "score" -> un entier qui vaut 1 si on like, et -1 si on dislike
    b. CommentLike contient : "$comment" -> référence vers un commentaire ; "$user" -> référence vers un user ; "score" -> un entier qui vaut 1 si on like, et -1 si on dislike
11. Permettre d'aimer ou non un post et un commentaire
    a. Ajouter 2 liens sur la page post.php et comment.php, pour "like" et "dislike"
    b. Persistez une nouvelle entité "PostLike" ou "CommentLike" quand on clique sur un de ces liens
    c. Vérifiez que ça fonctionne en regardant en base
12. Rajoutez une notion "d'amis" à vos entités, pour cela, créez un attribut "$friends" dans votre classe "User" qui a une relation vers la classe User (many-to-many self-referencing)
13. Ne filtrez dans les Posts affichés, que les votre et ceux de vos amis (Requête personnalisée, avec Repository et QueryBuilder) (rajoutez des entrées dans la table nouvellement crée pour vos tests)
14. Permettre l'ajout d'un ami :
    a. Lister les utilisateurs en base qui ne sont pas déjà vos amis
    b. Mettre un bouton: "ajouter à mes amis" qui enregistre cette nouvelle relation
15. Afficher le nombre de likes et dislikes par post (faire un COUNT avec le querybuilder, dans un répo personnalisé).
16. Si t'arrive là, déjà bien joué, et maintenant, faire une requête qui renvoie le nombre de likes par utilisateur (bonne chance!)