esprit Se former autrement HONORIS UNITED UNIVERSITIES	EXAMEN : **Module Technologies web 2.0** Semestre : 1 ■ 2 □ Session : Principale ■ Rattrapage □
Module: Technologies web 2.0 Enseignant(s): UP web Classes : 3A/4SE Documents autorisés : OUI □ NON ■ Nombre de pages : 5 pages Calculatrice autorisée : OUI □ NON ■ Internet autorisée : OUI □ NON ■ Date : 17/01/2022 Heure : 13h Durée : 1h30	

Nom et Prénom…… classe …….

Partie 1 : QCM (8 points)

Question	1	2	3	4	5	6	7	8
Réponse	D	C	B	C	B	D	C	D

Partie 2 : (12 points)

1- Quelles sont les étapes à suivre pour créer et configurer une base de données «AgenceImmo »? **[1pt]**

- Modifier le fichier .env (Database_url) [0.5 pt]
- Tapez la commande php bin/console doctrine:database:create [0.5 pt]

2- Lors de la création d'une entité via une commande quel(s) est (sont) le(s) fichier(s) que seront créés **[1 pt]**

- Un fichier php sous le dossier src\Entity [0.5 pt]
- Un fichier php sous le dossier src\Repository [0.5 pt]

3- La création de la clé étrangère sera effectuée dans quelle table ? **[0.5 pt]**

- Dans la table BienImmobilier

4- Corrigez le code de la figure ci-dessus pour que numprop soit la clé primaire de l'entité

```
/**
    * @ORM\Id
    * @ORM\Column(type="string", length=255)
    */
   private $numprop;

*/
```

propriétaire tout en gardant le fait que numprop est de type string. **[1 pt]**

5- Complétez le code nécessaire pour avoir une relation entre les deux entités comme indiqué dans le diagramme de classe **[1.5 pts]**

```
      /**

* @ORM\ManyToOne 0.25 pt(targetEntity=Proprietaire::class, inversedBy0.25 pt
="bienImmobiliers")

      */

private $numprop;
```

```
      /**
      * @ORM\OneToMany0.25 pt (targetEntity=BienImmobilier::class, mappedBy 0.25 pt
      =" numprop 0.5 pt")
      */

private $bienImmobiliers;
```

6- Complétez le code nécessaire pour avoir le formulaire correspondant à l'entité Bien-immobilier comme indique la figure suivante **[1.75pts]**

```
->add('numprop', EntityType 0.25 pt::class 0.25 pt,
        [
            'class 0.25 pt '=> Proprietaire 0.25 pt:: class 0.25 pt,
            'choice_label 0.25 pt '=>'nom 0.25 pt ',

    ]);
```

7- Complétez la fonction « deleteProp() » qui permet de supprimer un propriétaire et de faire une redirection vers la liste des propriétaires **[1.75 pts]**

```
/**
    * @Route("/deleteProp/{id}", name="deleteProp")
     */

public function deleteProp($id 0.25 pt)

{

$proprietaire =$this->getDoctrine()->getRepository() 0.25 pt (Proprietaire::class)-
>find 0.25 pt ($id);

    $em=$this->getDoctrine()->getManager() 0.5 pt;
    $em->remove($proprietaire)0.25 pt;
    $em->flush()0.25 pt;
return $this->redirectToRoute("PropList");
}
```

8- Q8

a) En tapant dans l'url http ://127.0.0.1 :8000/bien/immobilier/add, ce code a généré une erreur. Quelle est cette erreur sachant que les attributs dans la base de données n'acceptent pas une valeur nulle ? **[1 pt]**

- Erreur dans la base données la base n'accepte pas une valeur nulle

b) Corrigez ce code afin d'éviter cette erreur **[0 .5 pt]**

```
/**
 * @Route("/bien/immobilier/add", name="add_bien_immobilier")
 */
public function addBI(Request $request): Response
{
   $bienimmobiler=new BienImmobilier();
    $form1=$this->createForm(FormBienImmobilerType::class,$bienimmobiler);
    $form1->add('Ajouter',SubmitType::class);
    $form1->handleRequest($request);
if($form1->isSubmitted() && $form1->isValid())
        {
        $em=$this->getDoctrine()->getManager();
        $em->persist($bienimmobiler);
        $em->flush();
        return $this->redirectToRoute('r_list');

}

return $this->render(' bienimmobiler /add.html.twig', [
            'f' => $form1->createView(),
        ]);

    }
```

c) Modifiez la méthode addBI() sachant que dès l'ajout d'un nouveau bien-immobilier l'état soit par défaut 'disponible'.**[0.5 pt]**

```
/**
 * @Route("/bien/immobilier/add", name="add_bien_immobilier")
 */
public function addBI(Request $request): Response
{
  $bienimmobiler=new BienImmobilier();
   $form1=$this->createForm(FormBienImmobilerType::class,$bienimmobiler);
   $form1->add('Ajouter',SubmitType::class);
   $form1->handleRequest($request);
if($form1->isSubmitted() && $form1->isValid())
      {

$bienimmobiler->setEtat(«Disponible ») ;
      $em=$this->getDoctrine()->getManager();
      $em->persist($bienimmobiler);
      $em->flush();
      return $this->redirectToRoute('r_list');

}

return $this->render(' bienimmobiler /add.html.twig', [
        'f' => $form1->createView(),
      ]);
   }
```

9- Ecrire la fonction QueryBuilder « SortByDate() » qui permet d' afficher l'ensemble des biens immobiliers triés par leurs dates (ordre croissant) **[1.5 pts]**

On Accepte l'une de ces propositions :
DQL

```
public function SortByDate()
{
   $entitymanager=$this->getEntityManager();

$query=$entitymanager->createQuery('select b FROM App\Entity\BienImmobilier b order by b.date ');

return $query->getResult();

}
```

```
public function SortByDate()
{
        return $this->createQueryBuilder('b')

                ->orderBy('b.date', 'ASC')

                ->getQuery()

                ->getResult();

        }
```

QueryBuilder