Exemple 1

partie 1

qcm

1/ A-Response
2/D. Toutes les réponses sont correctes
3/C. php bin/console doctrine:database:create
4/ A. Index.php
5/D. Entité
6/C. Créer une indépendance du code vis-à-vis du SGBDR utilisé
7/D. Toutes les réponses sont correctes
8/D. Toutes les réponses sont correctes

partie2

1. Compléter le code pour que l'attribut experience puisse être nul (0.5pts)
[1] : `nullable: true`

```
/**
*@ORM\Column(type="string",length=255,[1])
*/
private $experience;
```

2. Complétez le code nécessaire pour afficher seulement les candidats avec spécialité Informatique. (1.25pts)

```
public function read(): [2]:Response
{$candidats = $this->getDoctrine()-> getRepository([3]Candidat::class)
->[4]findBy ([[5]"specialite"l=>"Informatique" [6]]);
return $this->render('candidat/read.html.twig', ['candidats'
=> $candidats]);
}
```

3. Complétez le code qui permet d'afficher les candidats comme le montre la figure suivante.
(1.5pts)

```
<h1> Liste des Candidats</h1>
<table border="1">
```

<tr>
<th>Cin</th>
<th>Nom - Prenom </th>
<th>Experience</th>
<th>Specialite</th>
</tr>
[7]{% for i in condidat %}
<tr>
[8]
<td>{{ i.cin}} </td>
<td>{{ i.nom | upper}} - {{i.prenom | upper}} </td>
<td>{{ i.experience}} </td>
<td>{{ i.specialite}} </td>
</tr>
[7]{% endfor %}

</table>

4. Pour chaque candidat, on souhaite ajouter un lien « planifier » comme le montre la figure
suivante :
Complétez le code qui permet d'ajouter le lien hypertexte, sachant que le lien nous redirige
vers la route nommée « planifiercandidat » et qui prend en paramètre l'id du candidat.
(1.25pts).
<a href="{{ path('planifiercandidat',{'id': i.id}) }}
">Planifier

5. En Cliquant sur le lien « planifier », le responsable va être redirigé vers l'interface de
planification d'entretien comme la montre la figure suivante :

```
->add('type', ChoiceType::class, [
    'choices' => [
        'Technique N1' => "TechniqueN1",
        'Technique N2' => "TechniqueN2",
```

```
        'Ressources Humaines' => "Ressources Humaines",
    ],
    'expanded' => true,
    'multiple' => false,
])
```

b. Complétez le code de la fonction qui permet de planifier un entretien à un candidat. (3.5pts)

```
/**
* @Route("add/{id}" , name =[18]entretien.add)*/
public function add(Request $request, $id , [20]ManagerRegistry $rep)
{
$e = [21] new Entretien();;
$c = $rep->getRepository(Candidat::class)->find($id);[22] ;
$e->setCandidat($c);
$e->setDate(new \DateTime()); // tjibelna date ta3 la7dhetha
$form = $this->createForm(EntretienType::class, $e);
$form->handleRequest($request);
if ($form->isSubmitted()) {
$em = $this->getDoctrine()->getManager();
$em->persist($e);
$em->flush();
}
return $this->render("add.html.twig",
["form" => $form->[24], "c"=>$c]);
}
```

6. Pour chaque candidat, on veut afficher les entretiens qui lui sont planifiés comme le montre
la figure suivante :

Complétez la fonction Query Builder qui permet d'afficher les entretiens spécifiques à un
candidat (2.5pts)

```
public function listByCandidat($id)
{return $this->[25] ('e')
->[26]join ('e.candidat', 'c')
->[27] addSelect('c')
```

```
->where('c.id=:id')
->setParameter('id',$id)
->[28]getQuery()->[29]getResult();
```