Compte-rendu
Épreuve E5

Pour l'implémentation du code, j'ai prévu de faire les missions dans l'ordre BAC, et c'est ce que j'ai fais.

I. Faire en sorte de pouvoir afficher sur une nouvelle page le détail d'une potion.

Premièrement, j'ai copier le fichier show.html d'un autre template et je l'ai adapté comme cela :

J'ai adapté le template à celui d'une potion, c'est a dire retirer les champs inutiles, les endpoints ont étaient remplacés par potion lorsqu'il y avait armure ou Armure (grâce au raccourci Ctrl+Alt+Maj+J) et rajouter les champs « Nom » « Description » et « soin ».

```html
<!DOCTYPE html>
<html lang="fr">
<head th:replace="mesFragments :: fragHead">
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<div class="min-vh-100">
    <nav th:replace="mesFragments :: fragNav"></nav>
    <main class="container">
        <h1>Détails de l'potion</h1>
        <div>
            <p><strong>Nom : </strong> <span th:text="${potion.nom}"></span></p>
            <p><strong>Description : </strong> <span th:text="${potion.description}"></span></p>
            <p><strong>Soin : </strong> <span th:text="${potion.soin}"></span></p>
        <div>
            <a th:href="@{'/admin/potion/'+${potion.id}+'/edit'}" class="btn btn-warning">
                Modifier
            </a>
            <a th:href="@{'/admin/potion'}" class="btn btn-primary">
                Retour à la liste
            </a>

        </div>
    </main>

</div>
<footer th:replace="mesFragments :: fragFooter"></footer>
</body>
</html>
```

Ensuite il a fallut rajouter la méthode show dans PotionControleur,

```kotlin
@GetMapping("/admin/potion/{id}")
fun show(@PathVariable id: Long, model: Model): String {
    val potion = this.potionDao.findById(id).orElseThrow()
    model.addAttribute("potion", potion)
    return "admin/potion/show"
}
```

Méthode qui a était copié d'un autre contrôleur, on a juste adapté le nom des variables et utiliser le dao de potion pour récupérer l'ID de la potion qu'on veut montrer.

II.Dans le tableau d'administration des « types d'armures » faire apparaître le nombre total de « type d'armures ».

Premièrement, j'ai afficher dans la vue de type d'armures donc « admin/typearmure/index » , une balise p pour afficher le nombre total de type d'armures :

```
<p>Nombre total de types d'armures : <span th:text="${nombreTotalTypesArmures}"></span></p>
```

Ensuite, tout simplement on utilise la méthode .size qui compte le nombre d'éléments d'une liste via le Contrôleur :

```
@GetMapping("/admin/typeArmure")
fun index(model: Model): String {
    // Récupère tous les types d'armures depuis la base de données
    val typeArmures = this.typeArmureDao.findAll()

    // Ajoute la liste des types d'armures au modèle pour transmission à la vue
    model.addAttribute("typeArmures", typeArmures)

    // Ajoute le nombre total de types d'armures au modèle pour transmission à la vue
    model.addAttribute("nombreTotalTypesArmures", typeArmures.size)

    return "admin/typearmure/index"
}
```

Ici, je récupère via le modèle le nombre de type d'Armure et ensuite je l'envoie à la vue grâce au contrôleur.

III.Lors de la création d'un personnage, faire en sorte que la liste des armures disponibles s'affichent et puisse être choisie.

Ici pour la vue j'ai juste rajouter une liste déroulante :

```
div class="mb-3">
    <label for="armure" class="form-label">Armure</label>
    <select class="form-select" id="armure" th:field="${nouvellePersonnage.armure}" required>
        <option th:each="armure : ${armures}" th:value="${armure.id}" th:text="${armure.nom}"></option>
    </select>
</div>
```

Je n'ai pas eu le temps mais je pense qu'il aurait fallut faire en sorte que le PersonnageControleur récupère via le modèle toutes les armures.