Rapport de Stage

Rapport de Stage Amir KATSIEV

L’entreprise :

J’ai effectué mon stage dans mon lycée Leonard de Vinci. Mon stage a été consacré à l’amélioration d’une application que mes professeurs m’ont confiée dans le cadre d’un projet. Mon objectif était de rendre l’application plus fonctionnelle, conviviale. Pour ce faire, j’ai effectué des recherches et des tests sur les différentes fonctionnalités de l’application, j’ai analysé les méthodes déjà implanté et j’ai travaillé avec mes professeurs pour identifier des éventuels problemès . J’ai également pu acquérir de nouvelles compétences en développement d’application et en collaboration d’équipe.

Besoin :

Pour répondre aux besoins du projet, mes professeurs nous ont confié un Trello pour organiser notre travail. Mon collègue de stage et moi avons également décidé de nous partager les tâches en fonction de nos compétences respectives. Nous avons également régulièrement communiqué et collaboré ensemble pour résoudre les problèmes qui se sont présentés tout au long du projet. Grâce à cette approche, nous avons pu répondre aux besoins du projet de manière efficace et professionnelle, en respectant les délais et les exigences de nos professeurs.

l’Environnement de travail

Pendant mon stage, j’ai travaillé avec un ensemble d’outils pour développer l’application. Nous avons utilisé SPRING BOOT un framework, également connu pour sa facilité d’utilisation, IntelliJ pour l’IDE, GitHub pour commit et push notre travail, et Kotlin pour le développement du code. Cela nous a permis de travailler avec efficacité.

Réalisations :

Voici mes 2 méthodes les plus intéressantes que j'ai peut effectuer durant ce stage.

- Export to Raw

```
@GetMapping("/exportRaw") ①
fun exportRaw(
@RequestParam("idTopic") idTopic: Long, ②
@RequestParam("idScope") idScope: Long,
request: HttpServletRequest,
model: Model
): ResponseEntity<String> {
val build = StringBuilder() ③
val questions = questionRepository.findByScopeIdAndTopicId(idScope, idTopic) ④
for (question in questions) { ⑤
build.append(questionService.questionToTextRaw(question)) ⑥
build.append("\n")
}

    val topic: Topic? = topicRepository.findById(idTopic).orElseThrow {
TopicNotFoundException("topic not found") } ⑦
    val fileName = "export-quizbe-${topic?.name}-${LocalDate.now()}.txt" ⑧
    return ResponseEntity.ok()
        .contentType(MediaType.TEXT_PLAIN)
        .header(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=$fileName")
        .body(build.toString()) ⑨
}
```

① Cette annotation répondra aux requêtes GET sur l'URL /exportRaw.

② @RequestParam prends les paramètres que cette méthode doit accepter. Dans ce cas, il s'agit de idTopic et idScope.

③ StringBuilder est une classe Java utilisée pour construire des chaînes de caractères. build sera utilisé pour stocker les questions exportées.

④ Cette ligne récupère toutes les questions associées à un idScope et idTopic.

⑤ Cette boucle itère à travers chaque question et les ajoute à build.

⑥ questionService.questionToTextRaw est appelé pour convertir chaque question en une chaîne de caractères formatée en texte brut.

⑦ ette ligne récupère le sujet associé à idTopic et lance une exception s'il n'est pas trouvé.

⑧ Cette ligne crée le nom de fichier pour le téléchargement.

⑨ Cette méthode renvoie un objet ResponseEntity contenant la réponse HTTP et le contenu texte des questions exportées.

- Export to Moodle

Cette méthode était la plus compliquer à concevoir, car il fallait déjà pour commencer à créer un cours sur Moodle et l'export en format XML et voir comment il est constitué pour pouvoir faire le même fichier avec notre méthode. j'ai utiliser la meme classe `SpringBuilder` qui permet de construire des chaines de caractères dans un fichier pour re produire le fichier XML de chez Moodle :

Voici un morceau du code :

```
val responses = responseRepository.findByQuestion(question)
val sumPositiveValue = responses.filter { it!!.value!! >= 1 }.sumOf { it!!.value!! }
val sumNegativeValue = responses.filter { it!!.value!! <= -1 }.sumOf { it!!.value!! }
var valueToGrade: Float = 0F
for (response in responses) {
if (response != null) {
if (response.value!! >= 1) {
valueToGrade = (response.value!!.toFloat() / sumPositiveValue.toFloat() * 100F)
} else
valueToGrade = (response.value!!.toFloat() / sumNegativeValue.toFloat() * -100F)
build.append("<answer fraction='")
build.append(valueToGrade)
build.append("' format='html'>")
build.append("<text>")
build.append("<![CDATA[")
```

- Probleme :

Le souci qu'on avait avec l'export de notre application vers Moodle, c'était de convertir nos values des questions en %, car les questions sur Moodle sont notées en pourcentage de -100% jusqu'à 100%, mais la note maximal de toutes les questions ne doit pas dépasser les 100%.

- Solution :

Nous créent 2 listes distinctes, une qui filtre les `value` positives et l'autre négatives des réponses de la question, ensuite avec la fonction `sumBy` nous faisons une somme des valeurs de la liste. Nous initialisons une variable `valueToGrade` de type Float (bien mettre un F après le 0 pour montrer que c'est bien un Float). Avec une condition, nous vérifions que si la `value` de la réponse est positive, alors nous prenons la réponse en la mettant en Float et divisons par la somme totale des `value` positives de nos questions et nous faisons une multiplication par 100. Cela nous permets d'avoir la `value` en pourcentage de toutes les `value` qu'on affecte dans la variable `valueToGrade` .

Bilan :

Je remercie mes professeurs pour m'avoir proposer ce stage. Cela m'a permis de mettre en pratique et d'améliorer mes connaissances en développement. J'ai aussi pu apprendre plein de nouvelles choses, notamment le framework Spring Boot ainsi que l'utilisation de GitHub, cela m'as permis de monter en compétence.