RAPPORT TECHNIQUE

Projet JEE

Manon Cheneviere Hugo Gilles François Renaud Baptiste Heard

Introduction

Dans le cadre de ce projet académique, nous avons été amenés à élaborer une plateforme web dédiée à la gestion du planning des épreuves des Jeux Olympiques 2024. Ce projet a été réalisé en utilisant la technologie Java Enterprise Edition (JEE).

Le présent document est structuré de manière à refléter notre méthode de travail tout au long du projet. Dans la première partie, nous examinerons en détail notre processus de réflexion initiale, les choix de développement que nous avons opérés, notamment le diagramme de classes et les spécificités de notre conception. Nous explorerons également les classes particulières essentielles à la gestion des utilisateurs et des sessions.

La deuxième partie de ce rapport abordera la question cruciale de la sécurité, soulignant l'importance des mesures mises en place pour garantir la confidentialité et l'intégrité des données. Nous mettrons en lumière l'utilisation de l'algorithme Bcrypt et les adaptations spécifiques apportées aux tables de routage pour renforcer la sécurité de la plateforme du mieux que nous avons pu au vu du temps imparti.

La troisième partie, dédiée au Frontend, se penchera sur les aspects visuels et ergonomiques de notre plateforme. Nous détaillerons notre réflexion initiale, mettrons en avant l'utilisation de Pebble en tant que moteur de template et discuterons de l'intégration de Bootstrap pour assurer une expérience utilisateur optimale.

Table des matières

Table des figures

I - Procédés d'élaboration

I - a - Réflexion

Pour réaliser l'architecture de notre application, nous avons tiré profit de notre travail sur l'UML. Nous avons de fait commencé par dessiner le diagramme de classes afin de nous répartir les tâches au mieux. Une moitié du groupe a de fait poursuivi le travail sur les diagrammes et l'autre moitié a démarré l'implémentation du back de l'application.

Afin de nous approcher au mieux d'une structure pérenne, nous sommes partis sur une architecture MVC (Model View Controller). A noter que nous avons ajouté des Repositories afin de gérer les accès en base de données. Ces classes permettent d'alléger les Models afin d'améliorer la lisibilité de notre code.

Les Controllers gèrent l'intermédiation entre les Views et les Repositories. Ils assurent le bon rendu des vues et préparent à la fois les données à envoyer en base de données depuis le front et celles à afficher dans celui-ci depuis la base de données.

Les Repositories contiennent les CRUD qui permettent l'envoi et la réception des données présentes en base de données et le mapping objet-entités et inversement. Cette classe nous permet de manipuler sereinement les données dont nous avons l'utilité.

Les Views permettent d'afficher les données que nous récupérons depuis la base, mais aussi de récolter les données entrées par les utilisateurs qui sont envoyées vers le Controller.

I - b - Choix de développement

I - b - 1 Diagramme de classes

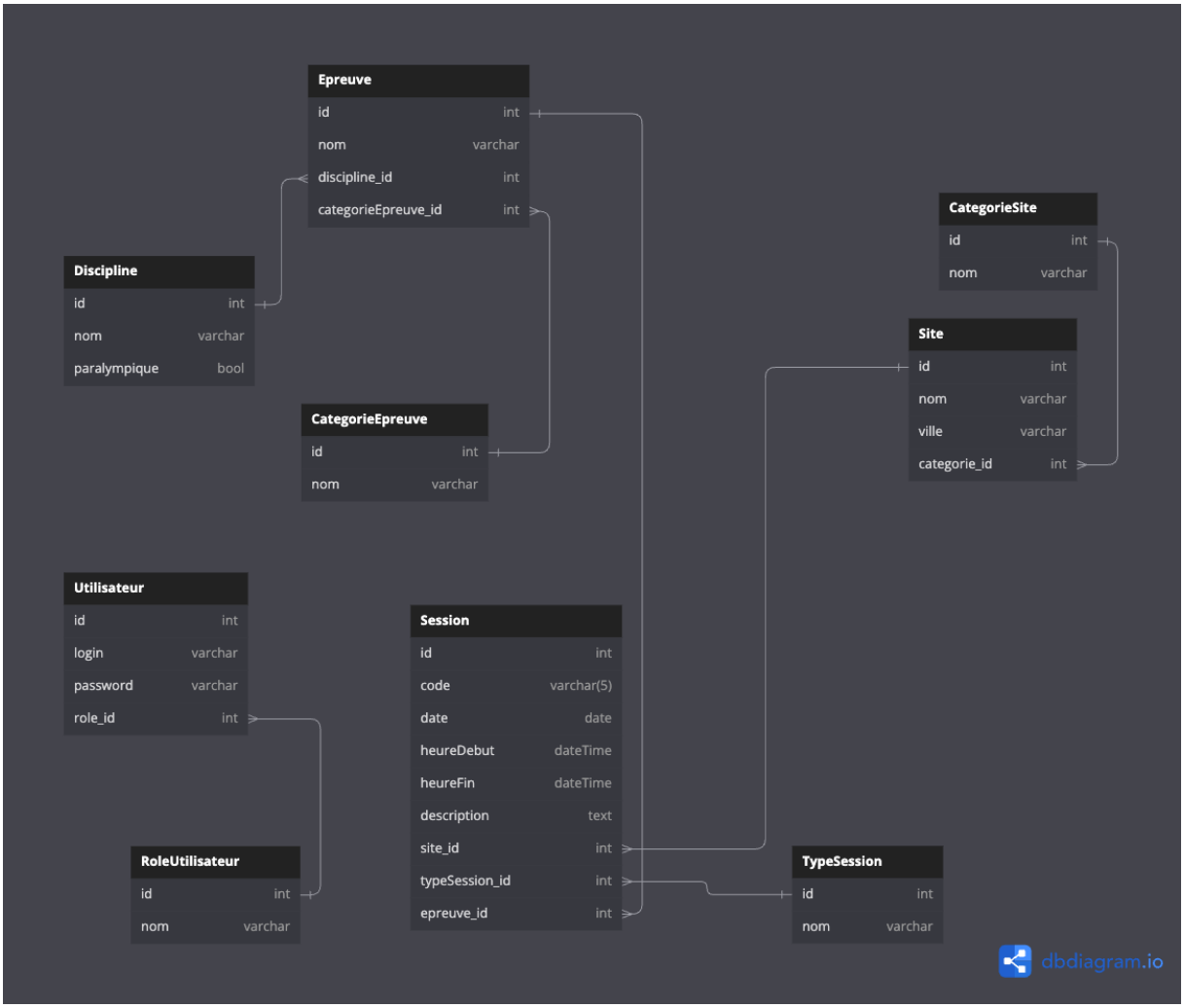

Figure 1 Diagramme de classes

I - b - 2 Spécificités

Nous avons fait le choix de réaliser le projet en JEE. De fait, nous avons considéré le temps disponible que nous avions et avons de concert estimé que celui-ci ne nous accordait pas suffisamment de marge pour réaliser le projet à l'aide de frameworks. En effet, la montée en compétence nécessaire était trop importante. Néanmoins grâce au TP préliminaire, nous avons pu clore ce projet en natif.

Ce problème s'est également répercuté sur la gestion des données. En effet, nous avons essayé de faire fonctionner un ORM avec Hibernate mais nous avons échoué. Nous avons donc rédigé les requêtes à la main. Cependant, leur nombre étant limité, cela ne nous a pas fait perdre énormément de temps.

Notre application dispose également de fonctions asynchrones, cela concerne tout particulièrement toutes les fonctions impliquant un accès à la base de données. Ce choix nous permet de réduire les temps de chargement, qui étaient auparavant conséquents.

I - c - Classes particulières

Chaque classe présente dans les modèles possède des CRUDs dans les Repositories ainsi que les méthodes imposées dans le sujet (statistique par exemple).

I - c - 1 Utilisateurs

Les attributs de la classe Utilisateur régissent l'utilisation de l'application dans son entièreté. En effet, le rôle détenu par l'utilisateur en ligne conditionne l'accès à certaines pages.

A noter que l'élaboration de la plateforme nous a poussés à modifier la structure de la base de données. Pour des soucis d'identification, nous avons ajouté les attributs nom et prénom.

I - c - 2 Sessions

Les sessions nous ont particulièrement posé souci notamment au niveau de l'affichage des dates. En effet, le passage entre le HTML et les objets Date n'était pas aisé.

II - Sécurité

Étant donné le délai que nous avions pour réaliser le projet et le fait que nous programmions en natif, nous avons fait au mieux pour ajouter des notions de sécurité dans l'application.

II - a - BCrypt

Pour encoder les mots de passe et ne pas les stocker en clair dans la base de données, nous avons utilisé l'outil BCrypt.

BCrypt offre une résistance élevée aux attaques par force brute et intègre automatiquement le salt dans le processus de hash.

II - b - Adaptation des tables de routage

Comme statué plus tôt, par souci de sécurité, nous avons conditionné l'accès aux différentes pages au rôle de l'utilisateur actif. Ainsi :

- Un administrateur a accès à toutes les pages et toutes les fonctionnalités
- Un gestionnaire de session a uniquement accès à la page sessions
- Un gestionnaire administratif a accès aux pages des sites, des disciplines et des épreuves
- Un gestionnaire administratif et de session a accès à toutes les pages sauf celle des utilisateurs
- Tous les utilisateurs ont accès à la page d'accueil.

III - Frontend

III - a - Réflexion

La représentation de nos données se fait principalement sous forme de tableaux dans des pages réservées pour chaque élément. Chaque élément dispose également de formulaires permettant de créer des éléments ou d'en modifier. A noter que les utilisateurs ont uniquement accès aux pages autorisées par leur rôle.

Ce layout reste sommaire car s'adaptant au temps que nous avions à disposition mais il permet d'avoir un visuel tout à fait satisfaisant sur les données manipulées.

-	Nom	Ville	Catégorie	Actions
13	Tour Eiffel	Paris	Stade	Modifier Supprimer
14	Vieux Lyon	Lyon	Salle de spectacle	Modifier Supprimer
15	Vieux-Port de Marseille	Marseille	Lieu public	Modifier Supprimer
16	Place de la Bourse	Bordeaux	Centre aquatique	Modifier Supprimer
17	Place du Capitole	Toulouse	Stade	Modifier Supprimer

Figure 2 Exemple de tableau de la page des sites

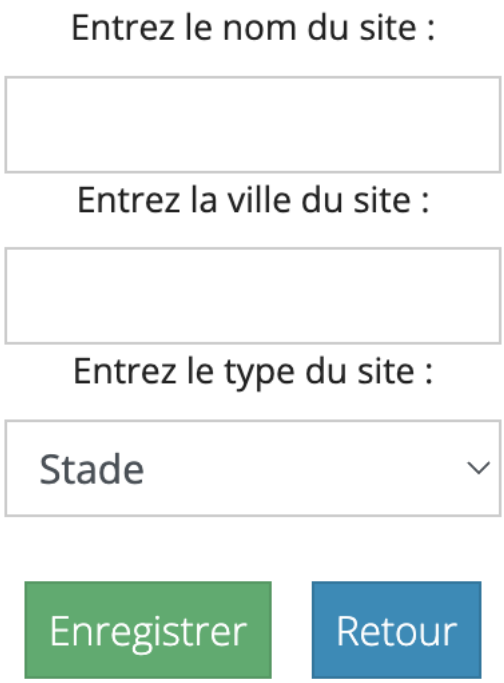

Figure 3 Exemple de formulaire HTML

III - b - Pebble

Afin de faciliter le développement des pages du front, nous avons utilisé l'outil Pebble, qui est similaire à l'outil Twig. Cet outil permet l'utilisation de boucles et de conditions directement dans le HTML par le biais de balises. Nous avons également pu injecter directement diverses variables depuis les Controllers.

Nos templates sont ainsi organisés par catégories de données (users, sessions, sites, épreuves) et chaque dossier contient à la fois les formulaires et les pages d'affichages des objets.

A noter que notre application contient une page base.html, de laquelle toutes les autres pages héritent. Cette page contient la barre de menu et les liens vers les feuilles de style.

```
<tbody>
    {% for epreuve in epreuves %}
        <tr>
         <td>{{ epreuve.id }}</td>
         <td>{{ epreuve.discipline.nom }}</td>
         <td>{{ epreuve.nom }}</td>
         <td>{{ epreuve.categorieEpreuve.nom }}</td>
    {% endfor %}
</tbody>
```

Figure 4 Exemple de code Pebble

III - c - Bootstrap

Pour rendre une application correcte en terme de design, nous avons utilisé Bootstrap. Pour ce faire, nous n'avons pas directement intégré la librairie dans le dossier lib mais nous avons inséré le lien vers Bootstrap dans base.html.

Cela nous a permis de styliser l'apparence de notre application (boutons, tableaux…).

Conclusion

En résumé, ce projet de plateforme web JEE pour la gestion du planning des épreuves des Jeux Olympiques de 2024 a été mené à bien grâce à une approche réfléchie au sein du groupe. La mise en place de mesures de sécurité telles que Bcrypt, les ajustements des tables de routage, ainsi que la conception d'une interface utilisateur intuitive avec Pebble et Bootstrap ont été les points clés de notre démarche. Ce rapport reflète notre engagement envers la qualité et l'efficacité de la solution, prête à répondre aux exigences spécifiques demandées.