Exercice 2 — PEL : Paris En Ligne

Nous voulons mettre en place un site web de paris en ligne sur les résultats de matchs sportifs. Le fonctionnement général du site web est le suivant :

- Un administrateur du site ajoute un match, avec le sport (foot, rugby,...), les noms des deux équipes qui doivent se rencontrer, la date et l'heure du match.
- Les joueurs peuvent alors parier sur le résultat final du match, en prédisant le vainqueur (ou match nul) et en misant une somme d'argent en euros.
- Les joueurs peuvent continuer à parier jusqu'à la dernière seconde avant le début du match, ensuite les paris sont clos sur ce match.
- Les joueurs peuvent aussi annuler leurs propres paris (ou un admin peut le faire) jusqu'au début du match.

Exemple : Le 16 juin 2018, à 12h00, match de foot France-Australie, pari de « Fred » sur un résultat match nul, 8€. (Attention, les données concernant les dates ont changé dans l'implémentation).

Une fois le match terminé, un administrateur saisit le résultat (équipe vainqueur ou nul) de ce match. Les gains sont alors répartis automatiquement entre les parieurs qui ont gagné (l'ensemble des mises sur le match moins 10% récupérés par le site sont répartis entre les gagnants en proportion de leurs mises).

Les utilisateurs, qui s'identifient en tapant leur login et mot de passe (on ne s'intéresse pas ici à la création des utilisateurs/administrateurs), peuvent voir la liste des matchs encore ouverts aux paris.

Dans cet exercice, nous nous intéressons à l'utilisateur classique qui peut parier sur un match ouverts et gérer ses paris. Vous n'avez pas à créer la partie *Administration* pour le moment.

Modélisation de l'application

La façade de votre modèle doit respecter l'interface suivante :

```
public interface FacadeParis {
    /**
     * Connecte un utilisateur à l'application, si le couple login/mdp est valide.
     *
     * @param login  le login de l'utilisateur qui se connecte
     * @param mdp    le mot de passe de l'utilisateur
     * @throws UtilisateurDejaConnecteException
     * @throws InformationsSaisiesIncoherentesException
     * @return l'utilisateur connecté
     */
    Utilisateur connexion(String login, String mdp)
        throws UtilisateurDejaConnecteException,
InformationsSaisiesIncoherentesException;
```

```
    /**
     * Retourne la liste des matchs sur lesquels on peut encore parier.
     *
     * @return les matchs qui ne sont pas encore commencés
     */
    Collection<Match> getMatchsPasCommences();

    /**
     * Ajoute un nouveau pari sur un match.
     *
     * @param login      le login de l'utilisateur qui veut parier
     * @param idMatch    l'id du match sur lequel porte le pari
     * @param vainqueur  le pronostic du vainqueur pour ce match ("equipe 1", "equipe
2", ou "nul")
     * @param montant    le montant parié
     * @return l'id unique du Pari enregistré
     * @throws MatchClosException si le match est déjà commencé ou fini
     * @throws ResultatImpossibleException si le vainqueur n'est pas correct (nul,
equipe 1 ou 2)
     */
    long parier(String login, long idMatch, String vainqueur, double montant)
        throws MatchClosException, ResultatImpossibleException,
MontantNegatifOuNulException;

    /**
     * Annule un pari existant sur un match.
     *
     * @param login    le login de l'utilisateur qui veut annuler son pari
     * @param idPari   l'id du pari à annuler
     * @throws OperationNonAuthoriseeException si l'utilisateur n'a pas le droit (pas
admin et ce n'est pas lui le parieur) ou si le match est déjà commencé/passé
     */
    void annulerPari(String login, long idPari)
        throws OperationNonAuthoriseeException;

    /**
     * Retourne un match existant selon son id.
     *
     * @param idMatch  l'id du match à retourner
     * @return le match
     */
    Match getMatch(long idMatch);

    /**
     * Retourne un pari existant selon son id.
     *
     * @param idPari  l'id du pari à retourner
     * @return le pari
     */
    Pari getPari(long idPari);
```

```
    /**
     * Retourne la liste des paris d'un utilisateur.
     *
     * @return tous les paris d'un utilisateur (terminés et en cours)
     */
    Collection<Pari> getMesParis(String login);

    /**
     * Ajoute un nouveau match. L'utilisateur doit être administrateur.
     *
     * @param login    le login de l'utilisateur souhaitant ajouter le match
     * @param sport    le sport de ce match
     * @param equipe1  le nom de l'équipe 1
     * @param equipe2  le nom de l'équipe 2
     * @param quand    la date à laquelle le match a lieu
     * @throws UtilisateurNonAdminException si l'utilisateur qui ajoute le match n'est
pas admin
     * @return l'identifiant unique du match ajouté
     */
    long ajouterMatch(String login, String sport, String equipe1, String equipe2,
LocalDateTime quand)
        throws UtilisateurNonAdminException;


    /**
     * Définit le résultat final d'un match, et distribue les gains aux parieurs.
L'utilisateur doit être administrateur.
     *
     * @param login     le login de l'utilisateur souhaitant définir le résultat du
match
     * @param idMatch   l'id du match
     * @param resultat  le résultat du match ("equipe 1", "equipe 2", ou "nul")
     * @throws UtilisateurNonAdminException si l'utilisateur qui ajoute le match n'est
pas admin
     * @throws ResultatImpossibleException si le resultat n'est pas correct (nul,
equipe 1 ou 2)
     */
    void resultatMatch(String login, long idMatch, String resultat)
        throws UtilisateurNonAdminException, ResultatImpossibleException;


    /**
     * Déconnecte proprement un utilisateur de l'application, selon son login.
     *
     * @param login  le login de l'utilisateur à déconnecter
     */
    void deconnexion(String login);


    /**
```

```
     * Retourne la liste de tous les paris présents dans l'application.
     *
     * @return les paris
     */
    Collection<Pari> getAllParis();

    /**
     * Retourne la liste de tous les matchs présents dans l'application.
     *
     * @return les matchs
     */
    Collection<Match> getAllMatchs();
}
```

1. Modélisez l'application sur une feuille, en respectant la nomenclature vue en cours. Vous ferez apparaître toutes les étiquettes de navigation, ainsi que les différentes variables attendues (avec leur scope).
2. Mettez à jour le fichier `pom.xml` afin de spécifier la verion de la JDK que vous utilisez, les dépendances nécessaires, et l'environnement d'exécution.
3. Développez les JSPs statiques et le contrôleur (servlet) permettant de gérer cette navigation, en vous inspirant des captures d'écran ci-dessous.

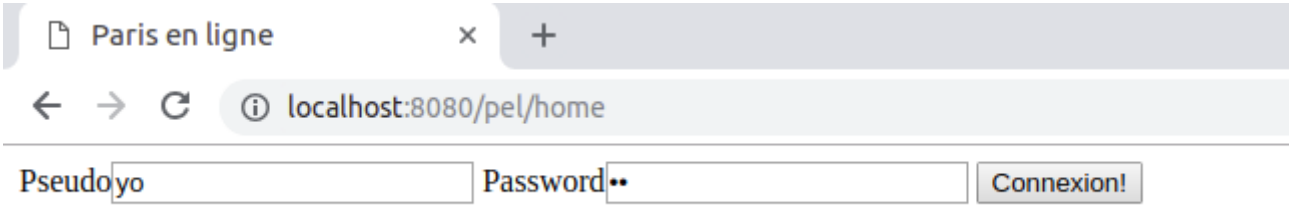


Figure 1. Page d'accueil

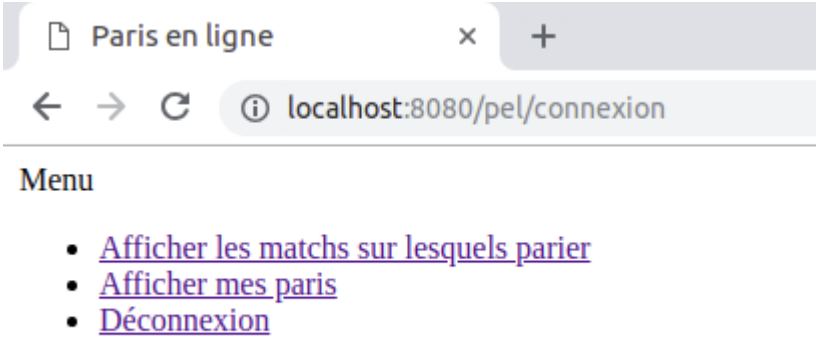


Figure 2. Menu principal

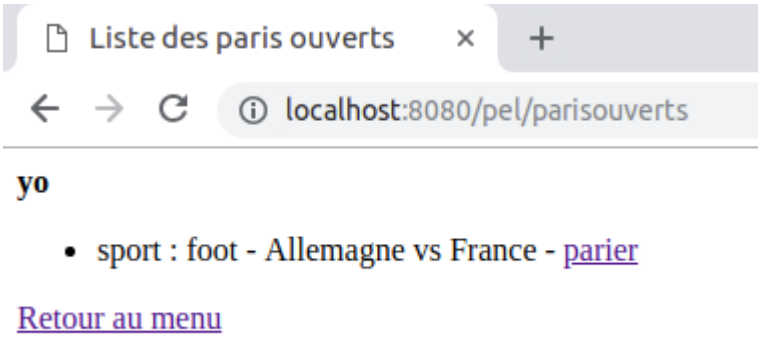


Figure 3. Matchs ouverts aux paris

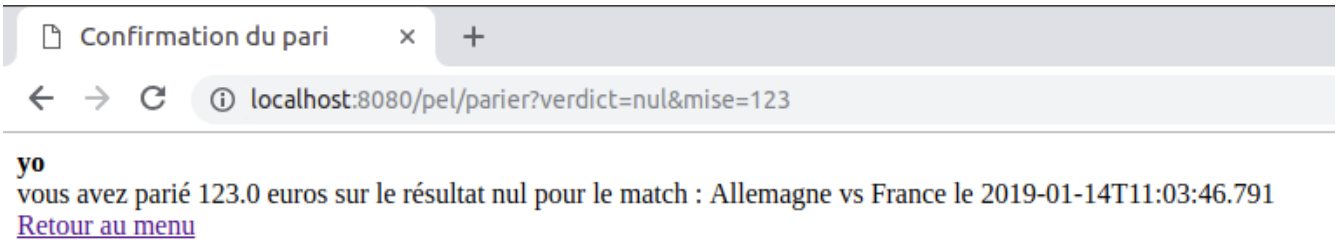


Figure 4. Pari confirmé

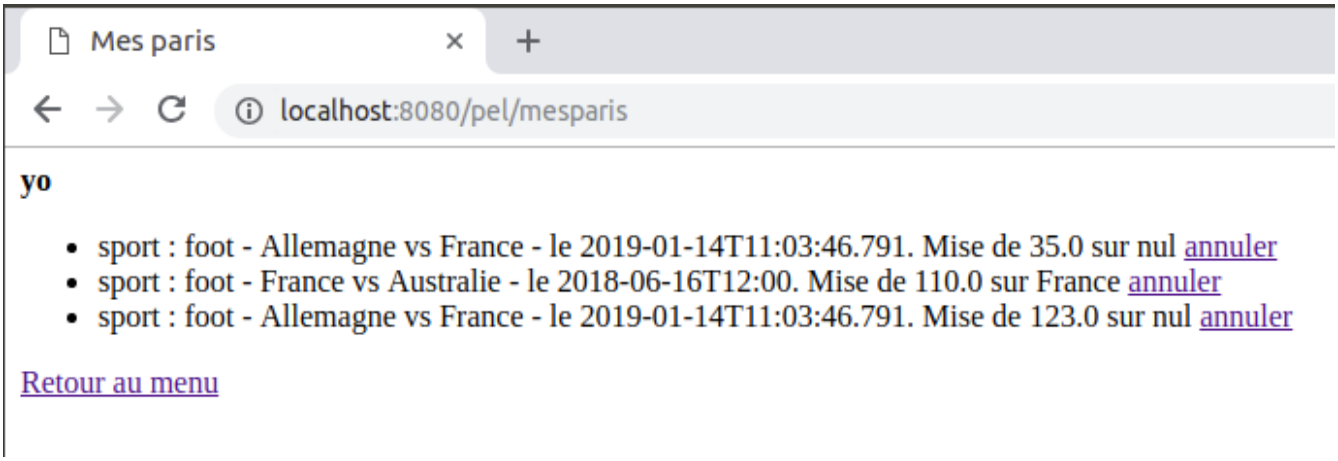


Figure 5. Gestion des paris

Figure 6. Confirmation annulation d'un pari

Injection des données — JSTL & Beans

À partir de maintenant, nous allons relier notre modèle/façade à notre application web. Nous allons pour l'instant traiter uniquement les scénarios sans traitement d'erreur.

Le contrôleur doit préparer les données à afficher dans les différentes JSPs, et il doit aussi effectuer

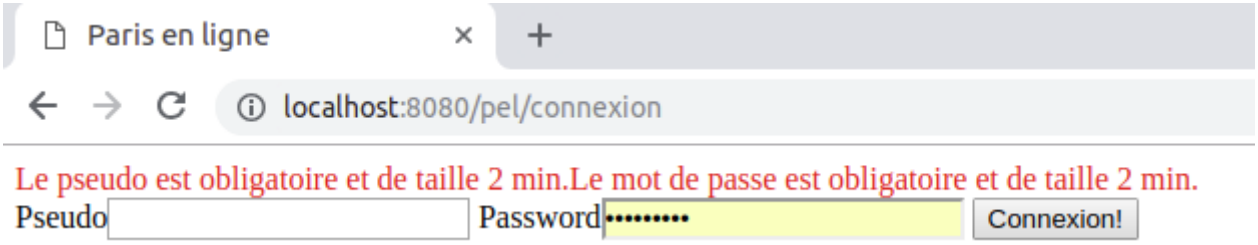
des traitements en interaction avec la façade. Par exemple, pour se connecter, un individu devra fournir un identifiant et un mot de passe. Le contrôleur vérifiera à l'aide de la façade si le couple des données est connu ou non.

1. Selon la modélisation que vous avez établie, effectuez les modifications nécessaires dans le contrôleur.
2. Mettez à jour les JSPs afin qu'elles puissent traiter les données et faire les bons appels au contrôleur. Par exemple, on veut être capable de sélectionner un match sur lequel parier. En cliquant sur ce match, le contrôleur devra être capable d'extraire l'identifiant du match concerné, afin de récupérer le match en question via la façade, et ainsi le préparer pour la prochaine JSP.

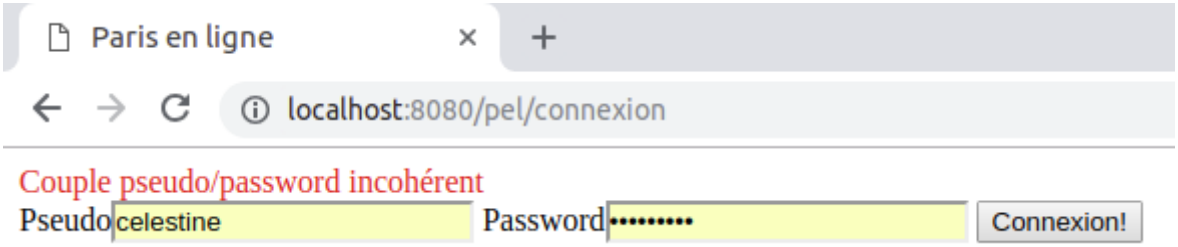
Gestion des erreurs

Ajoutez maintenant la gestion des différentes erreurs que l'on peut rencontrer dans l'application :

1. Erreur à la connexion (au moins un identifiant manquant)

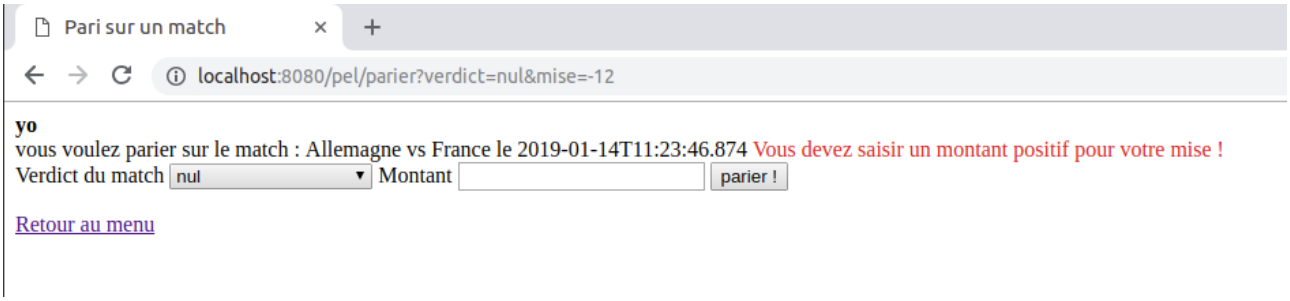


2. Erreur avec des identifiants inconnus par le SI



3. Erreur à l'annulation (le match est déjà fini et le pari ne peut pas être annulé)

4. Erreur à la saisie du pari avec un montant négatif.


Pari sur un match
localhost:8080/pel/parier?verdict=nul&mise=-12
yo
vous voulez parier sur le match : Allemagne vs France le 2019-01-14T11:23:46.874 Vous devez saisir un montant positif pour votre mise !
Verdict du match nul Montant parier !
Retour au menu