Rapport du Tp2-tp5

Partie JPA

Notre application est conçue pour gérer les tickets avec une architecture modulaire et extensible. Nous avons utilisé JPA pour la persistance des données. Les opérations CRUD sont encapsulées dans des DAOs et des services. Nous avons utilisé les énumérations pour normaliser les valeurs. Nous allons utiliser la branche Nana pour ce tp. Nous avons l'intention d'utiliser des différentes Branches pour chaque autre tp.

Entités Principales :

- Utilisateur :
  Représente un utilisateur du système.
  Attributs : id, nom, email, rôle.
  Relation : Auteur de Ticket, assigné à Ticket.
- Ticket :
  Représente un ticket dans le système de gestion.
  Attributs : id, titre, description, statut, priorité, date de création, date de mise à jour.
  Relations : Utilisateur (auteur), Utilisateur (assigné à), Commentaires, Tags.
- Commentaire :
  Représente un commentaire associé à un ticket.
  Attributs : id, contenu, date de création.
  Relation : Auteur (Utilisateur), Ticket.
- Tag :
  Représente une catégorie associée à un ticket.
  Attributs : id, libellé.

Énumérations :

- Statut : EN_ATTENTE, EN_COURS, RESOLU, FERME.
- Priorite : BASSE, MOYENNE, HAUTE.
- Role : UTILISATEUR, ADMINISTRATEUR, SUPER_ADMIN.

Packages DAO :

DAOs pour chaque entité : UtilisateurDao, TicketDao, CommentaireDao, TagDao, FicheDao.

Opérations CRUD pour chaque entité : création, mise à jour, suppression, récupération par identifiant.

Utilisation de l'EntityManager pour gérer les opérations de persistance.

Services :

Services pour chaque entité : UtilisateurService, TicketService, CommentaireService, TagService, FicheService.

Encapsulation de la logique métier dans les services.

Appel des DAOs appropriés depuis les services pour effectuer les opérations CRUD.

Classe JpaTest :

Classe de test pour démontrer l'utilisation des services et des DAOs.

Création d'instances d'entités, exécution d'opérations CRUD, gestion des transactions.

Utilisation de l'EntityManager pour la gestion des entités.

Exécution de la partie JpA :

- Démarrage du **run-hsqldb-server.bat** et **show-hsqldb.bat** car nous travaillons sur un ordinateur Windows.
- Exécuter **JpaTest** pour tester.

Partie 1 Servlet

Notre application web basée sur Java Servlets, utilisant les technologies Jakarta EE et HTML, il fournit des fonctionnalités de formulaire, de traitement de requêtes, et d'affichage de pages HTML.

Voici un récapitulatif global des composants de notre application :

Servlets :

- UserInfo Servlet :
  Cette servlet reçoit les données soumises par le formulaire HTML.
  Elle est responsable du traitement des données entrées par l'utilisateur, telles que le nom, le prénom et l'âge.
  Les données soumises peuvent être utilisées pour effectuer des actions spécifiques, telles que l'enregistrement dans une base de données ou la génération d'une réponse dynamique.
- accueilUser Servlet :
  Récupère les paramètres (nom, prénom, âge) d'une requête GET.
  Redirige l'utilisateur vers la page accueil.html en ajoutant les paramètres à l'URL.
- MyServlet :
  Répond aux requêtes GET et POST avec un message "Hello world" approprié.

Pages HTML :

- index.html :
  Affiche un message de bienvenue et une image de Firdaousse.
  Utilise des styles CSS pour la mise en forme.

- index2.html :
  Affiche un message de bienvenue et une image de Oumaima.
  Utilise des styles CSS pour la mise en forme.
- Formulaire HTML (myform.html):
  Permet à l'utilisateur de saisir son nom, prénom et âge. Lorsqu'il est soumis, les données sont envoyées à la servlet UserInfo.
- Page d'accueil utilisateur(accueil.html):
  Affiche un récapitulatif des informations fournies par l'utilisateur (nom, prénom, âge).
  Utilise JavaScript pour extraire les paramètres de l'URL et les afficher dans la page.

Exécution de la partie Servlet :

- Démarrage du **run-hsqldb-server.bat** et **show-hsqldb.bat** car nous travaillons sur un ordinateur Windows.
- Ouvrir un **PowerShel**l.
- Etre dans le répertoire courant du projet,
- Taper « **mvn compile jetty:run** » à la console.

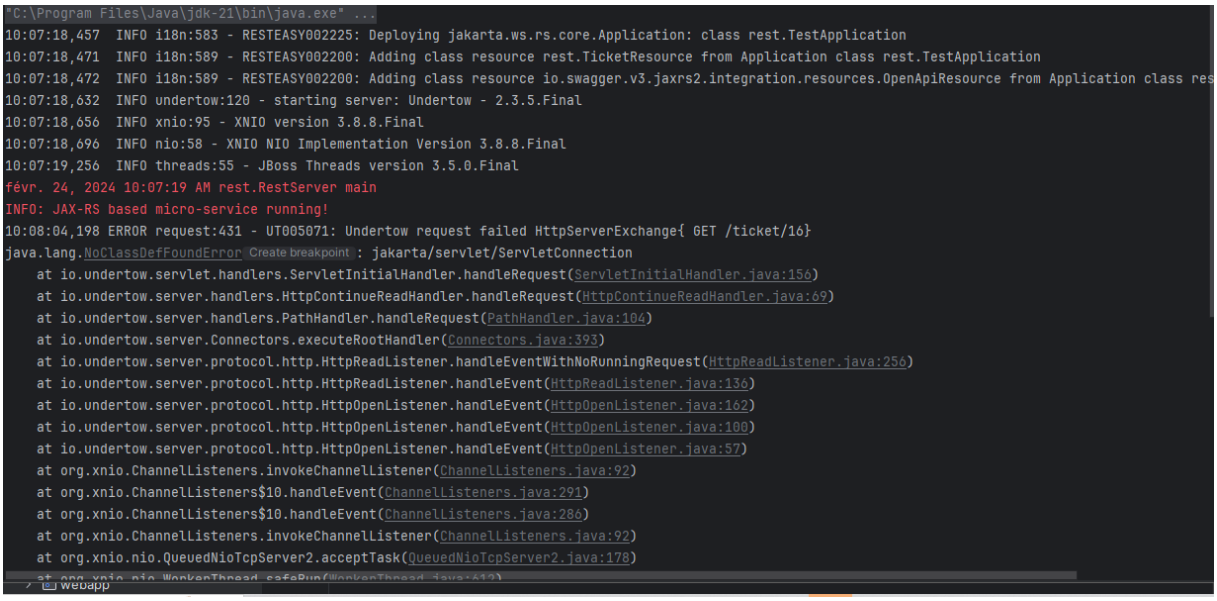
Partie 2 JaxRS et OpenAPI

En résumé, notre code met en œuvre un serveur REST avec des fonctionnalités pour gérer les tickets via des opérations REST définies dans TicketResource, ainsi que des fonctionnalités d'accès aux données avec JPA via IGenericDao et AbstractJpaDao. Il utilise des technologies telles que JAX-RS, JPA, Undertow pour fournir des services RESTful et accéder aux données de manière efficace.

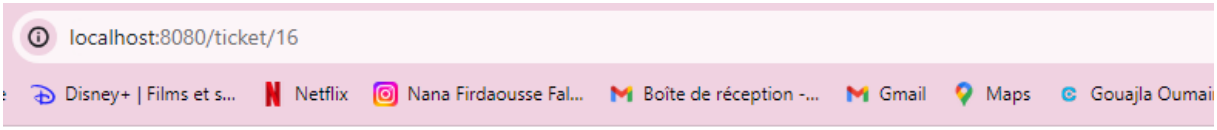
- Serveur REST (RestServer) :
  Initialise un serveur Undertow pour héberger le microservice RESTful.
  Utilise UndertowJaxrsServer pour déployer l'application REST.
  Démarre le serveur Undertow sur localhost:8080.
- Application JAX-RS (TestApplication) :
  Configure les ressources JAX-RS à déployer.
  Définit les ressources disponibles pour le serveur REST, notamment OpenApiResource et TicketResource.
- Ressource pour les tickets (TicketResource) :
  Définit les points de terminaison REST pour la manipulation des tickets.
  Utilise les annotations JAX-RS comme @Path, @GET, @POST, @PathParam, @Consumes, @Produces pour définir les opérations REST.
- Accès aux données avec JPA (IGenericDao et AbstractJpaDao) :
  Fournit une abstraction pour l'accès aux données avec JPA.
  IGenericDao définit les opérations CRUD de base.
  AbstractJpaDao fournit une implémentation de base pour les opérations CRUD en utilisant EntityManager.

Problèmes :

Notre serveur RestServeur est bien activé dans nous rencontrons quelques problèmes :



Lorsqu'on part dans http://localhost:8080/ticket/16



Notre serveur hsqldb-2.7.2.jar org.hsqldb.Server est bien activé et notre vpn activé .