**** institut supérieur de gestion**

**tunis le 14/12/2016**

***Rapport du projet JEE : Conception et réalisation d’une plateforme informatique pour la gestion d’une bibliothèque***

*travail élaboré par :jouini mohamed malek*

*3 lFIG 03*

*CIN: 07447773*

***Plan***

Énoncé du projet

Introduction

Diagramme de classe

Les outils

L’architecture

Création d'un projet EJB

Client lourd

Client web JSF

Android

1. **Enoncé du projet**

L’objectif du projet Architectures logicielles est de proposer une plateforme Informatique pour la gestion d’une bibliothèque.

Cette plateforme doit être capable de gérer 3 types de client:

1. Un client Léger (Web)

2. Un client Mobile

3. Un client lourd

**Le client léger doit être capable de :**

1. Rechercher des Ouvrages selon un ou plusieurs critères

2. Gérer ses préférences (Gestion du panier)

3. Passer des commandes

**Le client mobile doit être capable de :**

1. Rechercher des ouvrages selon un ou plusieurs critères

2. Passer des commandes

**Le client lourd doit être capable de:**

1. Gérer les ouvrages (Ajout, modification et suppression)

2. Gérer les prix et les promotions

1. **Introduction:**

Bibliothèque est une application faite en JEE qui s'adresse à plusieurs sortes d’utilisateurs. Ces derniers peuvent être de simples visiteurs passagers.

Chaque utilisateur peut rechercher des ouvrages (recherche simple ou avancée).

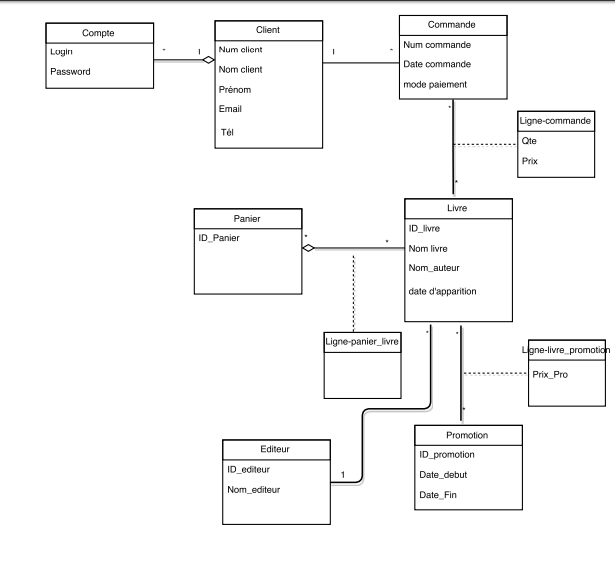
Les clients, après avoir passé par l’authentification, peuvent passer des commandes suite à leurs recherches.

\* les clients web peuvent également gérer leurs préférences.

\* le client lourd gère les ouvrages c est à dire ajoute, consulter, modifie et supprime des livre, et gère aussi les promotions, s’il y en a.

Il peut aussi gérer les comptes et les clients.

1. ***Diagramme de classe***

******

1. ***Les outils***

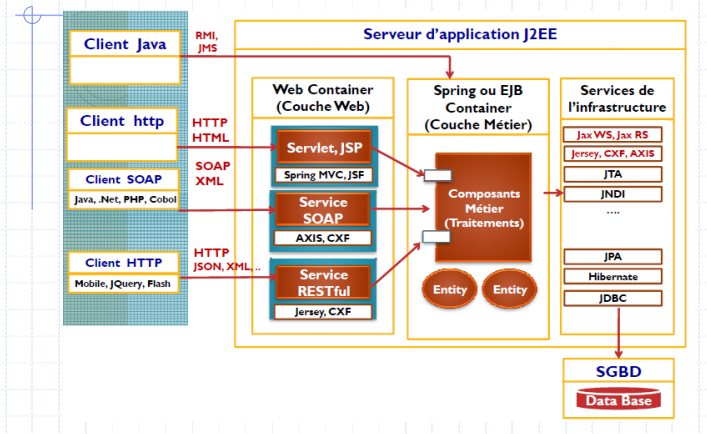
JDK

Eclipse Java EE IDE for Web Developers.θ

WildFly 8.0θ

EasyPhp

1. ***L'architecture***

******

1. ***Création d'un projet EJB***

Cette partie consiste en la création de la partie la plus importante du projet, nommé dans mon application Biblio et qui comporte le fichier persistence.XML et les trois types d'EJB :

**Entity Beans** (Les EJB entités): Représentent les données manipulées par l’application (Composants persistants). Chaque EJB Entity est associé à une table au niveau de la base de données

**Session Beans** (Les EJB session): Composants distribués qui implémentent les traitements du logique métier. Ces composants sont accessibles à distance via les protocoles RMI et IIOP.

**Service Beans :** service soap

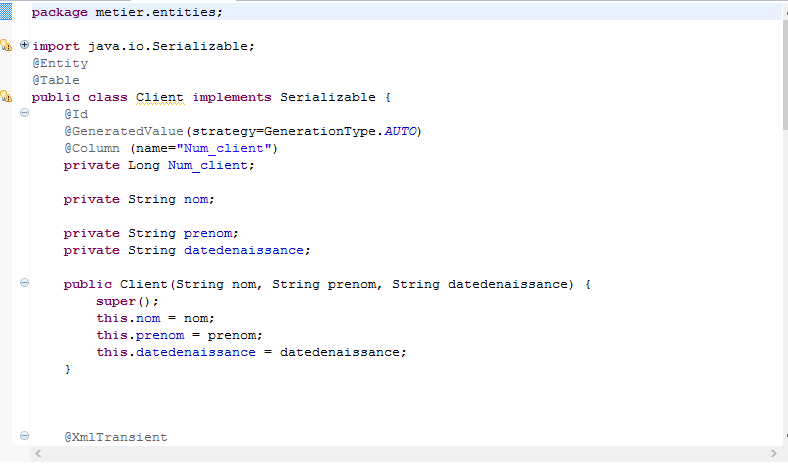
1. **Persistence.XML**

****

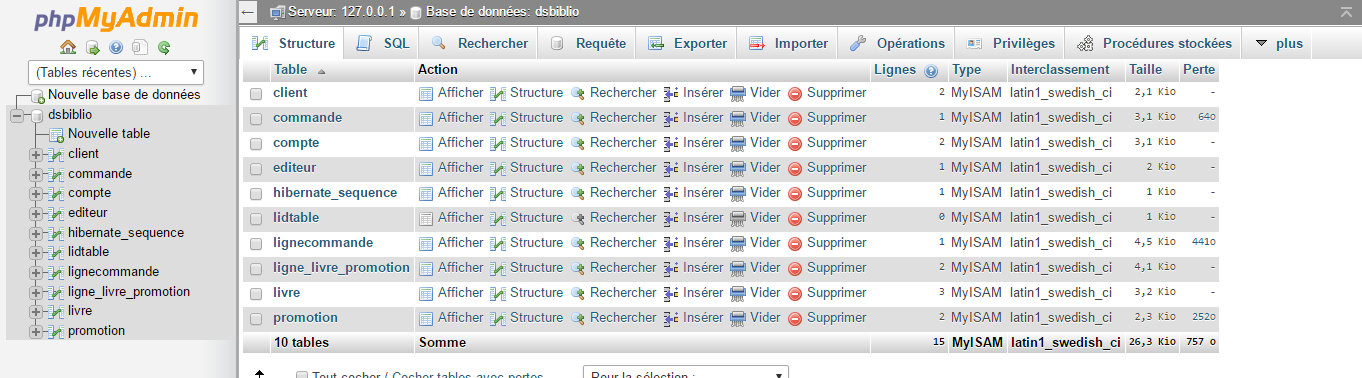
Permet de faire la connexion avec le data source.

1. **entity Beans:**

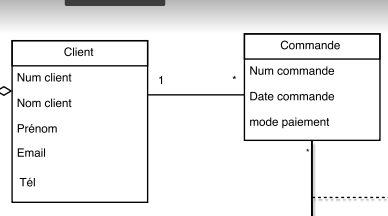
Exemple entité Client:



Pour qu'une classe puisse être persistente, il faut qu'elle soit identifiée comme une entité (@entity) , qu'elle possède un attribut identifiant en utilisant l'annotation @ID et qu'elle ait un constructeur sans arguments. Voici mes entités et mes classes associations insérées dans la base dsbiblio



**Les associations: OneToMany et ManyToOne:**

-

@OneToMany (mappedBy = "client")

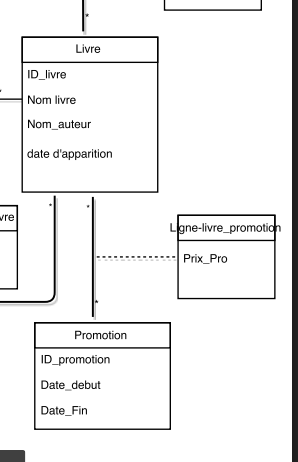
**private** Collection <Commande> commandes;

@ManyToOne

@JoinColumn (name="num\_client")

**private** Client client ;

***les associations ManyToMany***

******

@OneToMany

@JoinColumn (name="ID\_Livre")

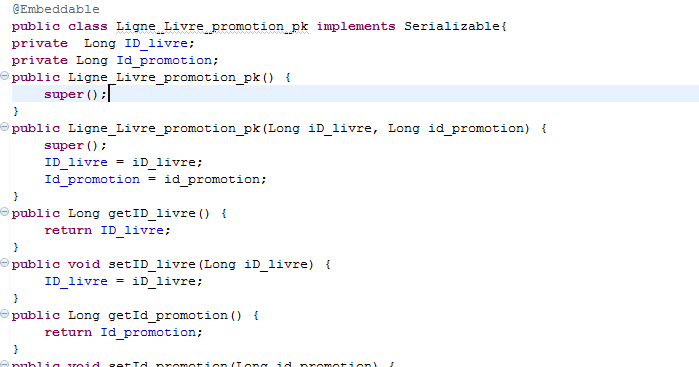
**private** Collection <LigneLivrePromotion> lignelivrepromotion;

@OneToMany

@JoinColumn (name="id\_promotion")

**private** Collection <LigneLivrePromotion> lignelivrepromotion;

Ensuite on crée la classePK LigneLivrePromotionPK qui contiendra les 2 clés primaires des tables associées livre et promotion comme suit

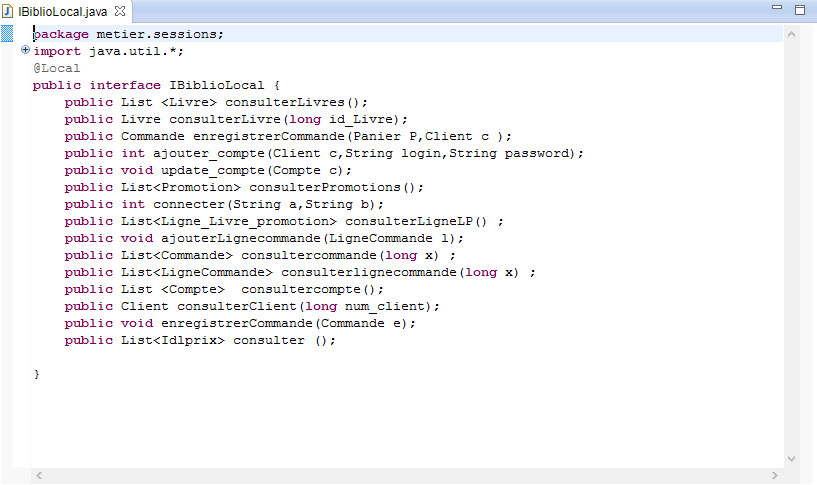
Par la suite , on implémente la classe LigneLivrePromotion qui appellera la classe LigneLivrePromotionPK comme une clé primaire de sa classe.

1. **sessions Beans**

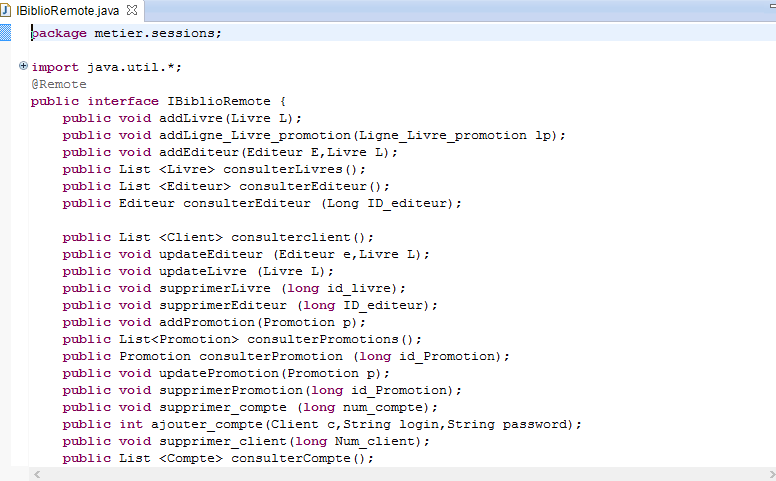
metier.sessions comporte 2 interfaces IBibliothequeRemote et IBibliothequeLocal et une classe classeEJBimplementation.

cette dernière implemente les méthodes de ces deux interfaces.

**IBibliothequeLocal** qui permet de déclarer les méthodes qui sont accessible en local, à travers cette interface, le client léger peut consulter les ouvrages ainsi que les promotions,

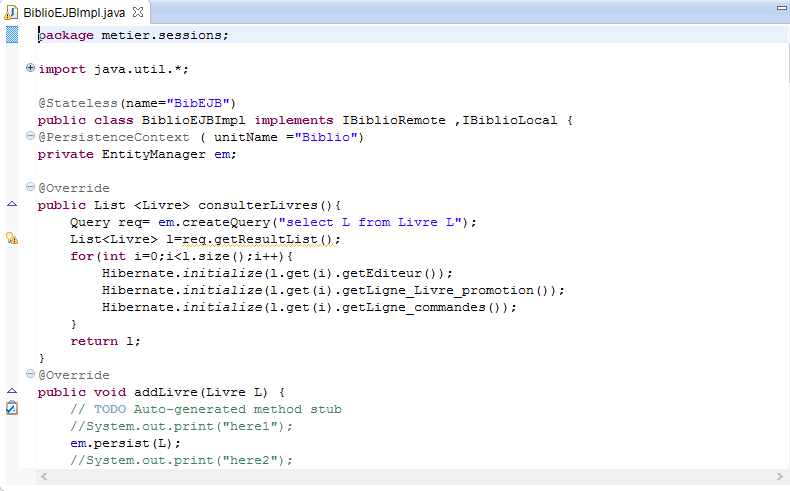


**IBibliothequeRemote :** qui permet de déclarer les méthodes qui sont accessibles à distance, à travers cette interface, le client lourd gère les ouvrages, les éditeurs, les comptes, les clients et les promotions.



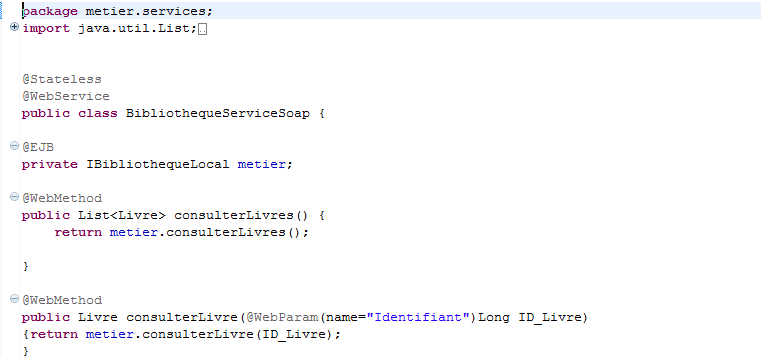
**classeEJBimplementation:** classe du Bean qui implémente les deux interfaces remote et local. L’implémentation des

Méthodes de cette classe représentent les traitements métier de l’application.

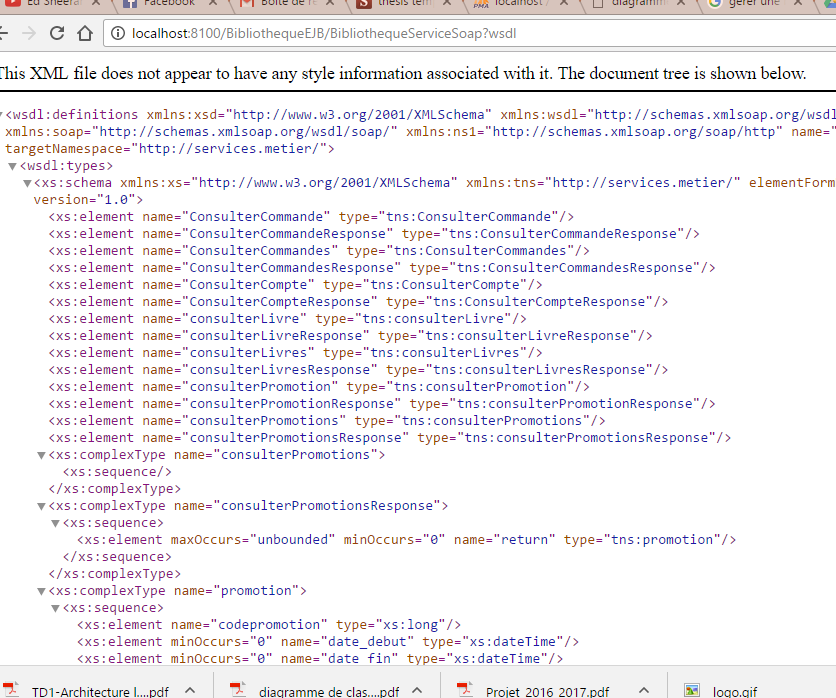


1. ***Services Beans***

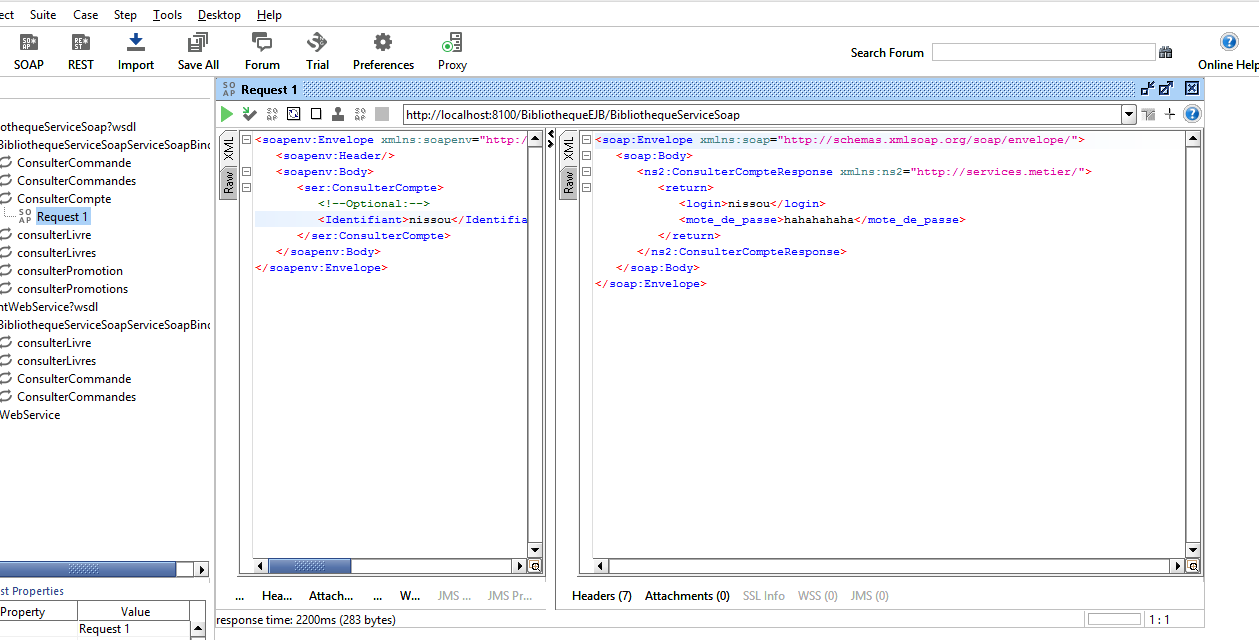
SOAP est un protocole d'invocation de méthodes sur des services distants. Basé sur XML, et a pour principal objectif d'assurer la communication entre machines.



La consultation du service Soap se fait WSDL comme suit:



Et voici un exemple sur son exécution sur SOAP



1. ***Client lourd***

J'ai commencé par créé un projet java que j'ai nommé ClientlourdEJB auquel j'ai ajouté le projet Biblio via Java Build Path ensuite j'ai créé les interfaces tel que gestion livres , gestion promotions; gestion clients et gestion clients en utilisant Jframe.

Voici un exemple de l'interface gestion Client avec l'exécution de la fonction ajout



1. ***Web Service Rest***

REST est un style d’architecture permettant de construire des applications (Web, Intranet, Web Service). Il s’agit d’un ensemble de conventions et de bonnes pratiques à respecter et non d’une

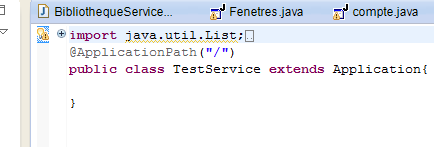
Technologie à part entière.

L’architecture REST utilise les spécifications originelles du protocole HTTP, plutôt que de réinventer une surcouche (comme le font SOAP ou XML-RPC par exemple).

j'ai commencé par créé un projet web dynamique dans lequel j'ai créé une classe java que j'ai nommé ServiceRest; en voici un aperçu



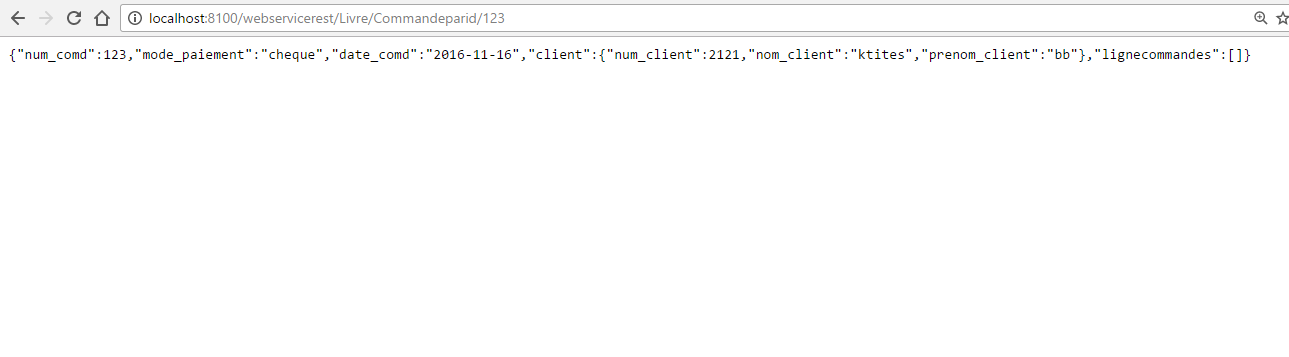
et une classe TestService



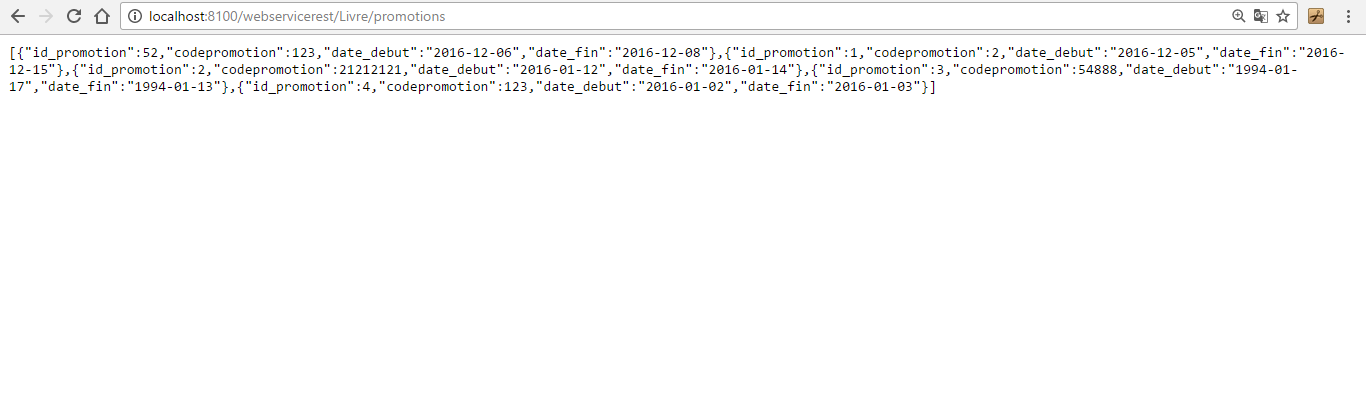
Et pour le déploiement, j ai créé un projet Ear pour lier ce service rest au projet Bibliot.

voici des exemples d'exécution rest:

consulter commande par id



ou encore consulter toutes les promotions

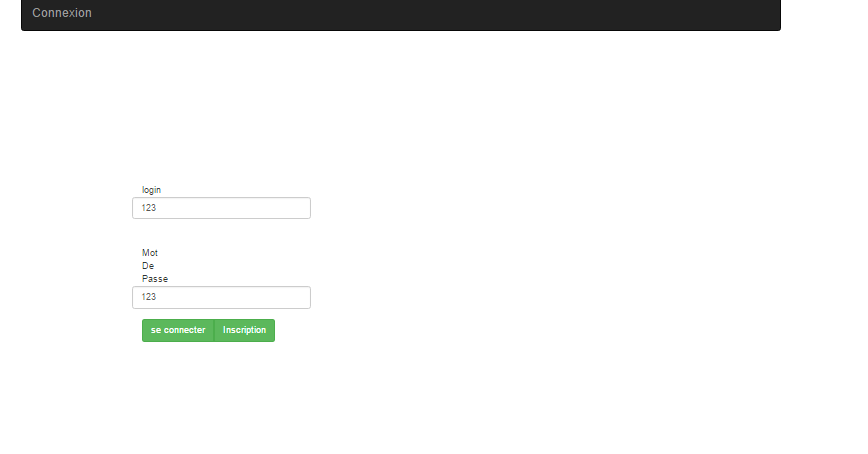


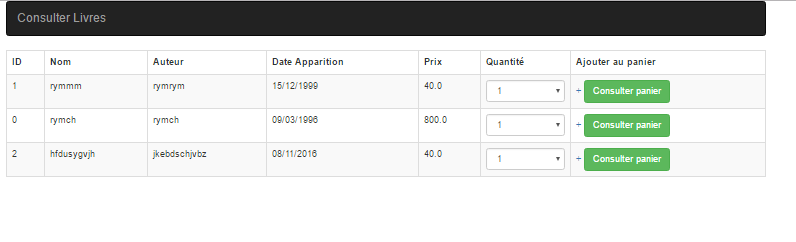
1. ***Client Web JSF***

J'ai créé un projet web dynamique dans lequel j'ai créé des classes java bean qui sont LivreMB , inscriptionMB Et livreprixqteMB,loginMB,livreqte,promolivreprix chacune lié à une Vue biensur lier ce projet avec le Biblio par le EarJSF.

Voici un exemple d'un managed bean:

Ainsi que la vue associée





1. ***Android :***

Pour la partie Android j’ai créé un Android application Project contenant 11 classes

