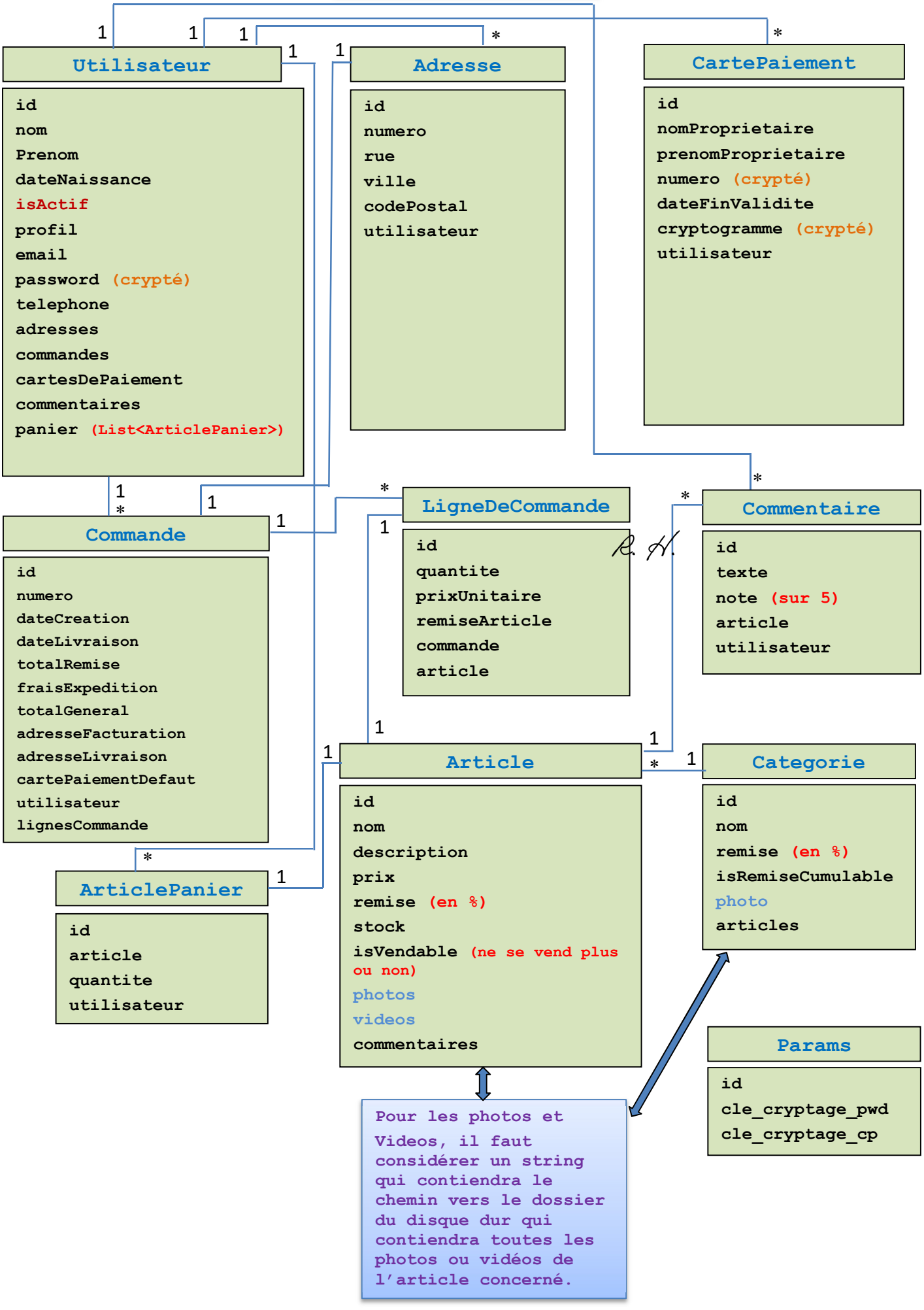
R. H. R. H.

Diagramme des Entités



R. H. R. H.

Relation entre la table *article_panier* et la table *article* :

Table : article_panier

id	article_id	quantite	utilisateur_id
…	…	…	…
…	…	…	…
…	67	3	10
…	88	2	10
…	67	1	55
…	90	5	55
…	106	2	55
…	…	…	…
…	…	…	…
…	…	…	…

Table : article

id	nom	description	prix	…	…
…	…	…	…	…	…
…	…	…	…	…	…
67	…	…	…	…	…
…	…	…	…	…	…
88	…	…	…	…	…
…	…	…	…	…	…
90	…	…	…	…	…
…	…	…	…	…	…
106	…	…	…	…	…
…	…	…	…	…	…

I. Considérations Générales :

Il faudra considérer 3 profils différents d'un utilisateur, il s'agit de : Client / Magasinier / Admin.

Il faudra donc créer une énumération pour utiliser ces trois profils, ce qui permettra plus tard de rajouter un nouveau profil facilement.

Lorsqu'un utilisateur se connecte et il a le profil "Magasinier" alors il est automatiquement redirigé vers la page gestion-articles.xhtml
Dans cette page, il va pouvoir ajouter/modifier/supprimer des articles à la base de données.

Ensuite, lorsqu'un client se connecte et il a donc le profil "Client", il sera redirigé vers la page gestion_chats.xhtml
Dans cette page il verra la liste des catégories et, en cliquant sur une catégorie il verra la liste des articles vendables et disponibles, et devant chaque article il y a une case à cocher. Une fois il coche un ou plusieurs articles et il clique sur ajouter au panier, ce dernier verra son nombre mis à jour.

Ensuite quand il clique sur le bouton "**valider le panier**" il sera redirigé vers la page "commande.xhtml" qui affiche le résumé des articles et avec le bouton "**Valider la commande**".

Une fois il clique sur "**Valider la commande**", alors il sera redirigé vers la page de paiement en affichant les cartes de paiement où il sélectionnera une, ensuite il clique sur le bouton "**Valider le Paiement**".

Le cryptage à utiliser pour le « mot de passe » utilisateur et le « numéro de carte » de paiement sera celui de **l'algorithme AES** avec **56 bits (multiple de 8)**, pour tous les utilisateurs.

La clé de cryptage concernant les cartes de paiement et celle concernant le mot de passe utilisateur doivent être créées au préalable (via une méthode main séparée) dans la table **Params**.

1. Toutes les pages html doivent utiliser un template qui découpe la page en trois parties :

- **En-tête :** affiche le logo, le nom du site, et les infos suivantes selon le cas :
  - Si l'utilisateur est connecté :
    - Le nom de l'utilisateur connecté (« Bonjour *Prénom* »),
    - Le lien de *déconnexion* (redirige vers login-utilisateur.xhtml),
    - Le lien du *panier* (redirige vers panier.xhtml) avec, à côté (droite ou gauche), le nombre d'articles contenus dans le panier,
    - Le lien vers l'historique des commandes (redirige vers commandes.xhtml).
  - Si l'utilisateur est déconnecté :
    - Le lien de *connexion* (redirige vers login-utilisateur.xhtml).

- **Corps :** affiche la page courante en fonction du profil dans le cas où un utilisateur est connecté, sinon affiche la page de connexion ou la page de création de compte client.
- **Pieds de page :** les infos de contact, le Copyright, … etc.

Pages xhtml à créer (côté FRONT-END) :

1) index.jsp : qui redirige systématiquement vers accueil.xhtml
2) accueil.xhtml
3) add-utilisateur.xhtml
4) login-utilisateur.xhtml
5) article.xhtml
6) gestion-achats.xhtml
7) gestion-articles.xhtml
8) gestion-admin.xhtml

II. Fonctionnalités à réaliser de bout en bout selon le profil :

1. Profil « Client » :

1.1. Page de « accueil » : (*accueil.xhtml*)

Cette page doit proposer deux liens de redirection, un pour créer un compte utilisateur (redirige vers add-utilisateur.xhtml) et un autre pour se connecter (redirige vers login-utilisateur.xhtml)

- Page add-utilisateur.xhtml : Permet la Création d'un compte utilisateur. Pour cela, il devra pouvoir saisir :
  - Les infos personnelles (voir les paramètres de l'entité Utilisateur)
  - Une ou plusieurs adresses, en indiquant celle utilisée par défaut pour la facturation (*adresseFacturationId*) et celle par défaut pour la livraison (*adresseLivraisonId*)
  - Une ou plusieurs cartes bancaires, en indiquant celle utilisée par défaut pour le paiement de ses achats (*cartePaiementDefautId*).
- Page login-utilisateur.xhtml : Permet de Se connecter avec ses email et mot de passe :
  - Si l'utilisateur n'est pas reconnu (email et/ou mot de passe incorrects), alors afficher une erreur sur la même page en lui donnant toujours la possibilité de se connecter ou de créer un compte utilisateur.
  - Si l'utilisateur est reconnu, alors on le redirige vers la page des gestion-achats.xhtml.

1.2. Lien de « Déconnexion » :

Lorsqu'on un utilisateur est connecté et qu'il clique sur le lien de déconnexion, alors cela devra avoir pour effet :

- La suppression du nom de l'utilisateur connecté dans l'entête de la page
- La suppression du lien vers l'historique des commandes
- La remise à zéro du panier (vider le panier)
- Puis La redirection vers la page de connexion à savoir login-utilisateur.xhtml.

1.3. Page des « Achats » : (*gestion_chats.xhtml*)

Cette page doit afficher dans son entête :

- Un lien permettant à l'utilisateur de se déconnecter,

- Un lien permettant d'afficher la page « Panier » et qui affiche en même temps à côté le nombre d'articles déjà ajouté au panier.
- Un lien permettant d'afficher l' «Historique des commandes ».

Cette page doit afficher dans son corps :

- La liste des catégories et les articles par Catégorie : En sélectionnant une catégorie, la liste des articles détaillés doit s'afficher automatiquement avec la possibilité d'ajouter un article au panier avec une quantité à saisir….etc.

1.4. Page du « Panier » : (*panier.xhtml*)

Cette page doit afficher :

- La liste de tous les articles du panier avec leurs détails (nom de l'article, brève description, prix unitaire, quantité modifiable, remise en %, total remise, prix total), bouton ou lien de suppression de l'article du panier, le total général.
- Le bouton « **Valider le Panier** » qui renvoie vers la page « *commande.xhtml* ».

1.5. Page de la « Commande » : (*commande.xhtml*)

Cette page doit afficher :

- Le détail de toutes les lignes de commande : chaque ligne de commande correspond au détail d'un article à savoir nom, brève description, quantité, remise, total. En plus de cela, rajouter une case à cocher qui permettra d'indiquer si l'article fera partie de la commande ou non.
- Les frais d'expédition
- Le total général
- La carte de paiement par défaut
- L'adresse de facturation et l'adresse de livraison par défaut
- Le bouton de validation finale pour procéder au paiement.

En cochant une des cases des lignes de commande, cela devra mettre à jour le total général

En cliquant sur le bouton « Valider la commande », cela devra effectuer les opérations suivantes :

- Prendre en compte uniquement les lignes de commandes cochées.
- Créer la commande
- Mettre à jour le stock des articles
- Mettre à jour le panier
- Redirection vers la page « Commande réussie » (*commande-ok.xhtml*) où l'on affichera un bouton de retour vers la page des « Achats ».

1.6. Page de « Historique des Commandes » : (*historique-commandes.xhtml*)

Cette page doit afficher la liste de toutes les commandes du client connecté.

En cliquant sur une commande, cela devra afficher le détail complet d'une commande.

1.7. Autres Pages (à définir selon le besoin) : (*XXXXX.xhtml*)

…

2. Profil « magasinier » :

Lorsque le magasinier se connecte, il est redirigé automatiquement vers la page « *gestion-achats.xhtml* »

Il doit pouvoir par exemple : créer/modifier/supprimer un article.

3. Profil « Admin » :

Lorsque l'administrateur se connecte, il est redirigé automatiquement vers la page « *gestion-admin.xhtml* ».
L'administrateur gère :

- La création/Modification/Suppression d'un magasinier
- La création/Modification/Suppression d'un administrateur
- La création/Modification/Suppression d'un client WebService

Un client WebService est un client externe (une application) qui peut effectuer des appels aux WebServices en indiquant un login et un mot de passe car les web services sont sécurisés.

III. Fonctionnalités des WebServices :

1. WebService REST « *getCommentaires(Integer idArticle)* » :

Développer un WebService accessible par tout le monde, donc sans authentification, qui permet de renvoyer une réponse contenant les commentaires d'un article donné, au format d'échange JSON.

Dans l'objet **Response** il y a, entre autres, l'objet « **List< Commentaire>** ».

2. WebService REST « *getCommentaires()* » :

Développer un WebService avec accès restreint (seul le profil "WSC" [Web Service Commentaire] et avec une authentification par login et mot de passe est autorisé), qui permet de renvoyer une réponse contenant tous les commentaires de tous les articles, au format d'échange JSON.

Sont concernés uniquement les articles qui ont au moins un commentaire.

Dans l'objet **Response** il y a, entre autres, l'objet « **Map<Article, List<Commentaire>>** ».

3. WebService REST « *getNbCommandesParVille()* » :

Développer un WebService accessible par tout le monde, donc sans authentification, qui permet de renvoyer une réponse contenant les nombres de commandes passés par ville, au format d'échange JSON.

Dans l'objet **Response** il y a, entre autres, l'objet « **Map<String, Map<Integer, Integer>>** ».

4. WebService REST « *getUtilisateursAvecPanierNonVide()* » :

Développer un WebService avec accès restreint (seul le profil "WSM" [Web Service Marketing] et avec une authentification par login et mot de passe est autorisé), qui permet de renvoyer une collection d'utilisateurs et, avec pour chacun, la liste des articles de son panier (ne sont concernés que les utilisateurs ayant un panier non vide).
Le but de ce WebService est de pouvoir exploiter ces données afin d'envoyer aux utilisateurs concernés des emails contenant la liste des articles du panier afin de les inciter à acheter (exemple : « Vous avez des articles en attente d'achat … »).

Dans l'objet **Response** il y aura donc, entre autres, l'objet « **Map<UtilisateurDto, List<ArticleDto>>** »), au format d'échange JSON, où :

L'objet **`UtilisateurDto`** est semblable à un **POJO** et sera constitué des attributs suivants :

`nom / prenom / email / telephone / panier`

L'objet **`ArticleDto`** est semblable à un **POJO** et sera constitué des attributs suivants :

`id / nom / description / photo` (⇒ correspond à la 1ère photo de la liste des photos de l'article concerné)

Voici les interfaces et classes à créer :

Couche Frontend (Backing Beans) :

UtilisateurBean / GestionaAchatBean / GestionArticlesBean / GestionAdminBean

- Couche Backend (Interfaces et classes) :

- Couche persistance (Interfaces et classes) :