POLYTECH GRENOBLE

# CONCEPTION SYSTEME

WeDressYou

Laissez-nous faire !

**ECOM 2015-2016**

**Hamdani Youcef – Mesnier Vincent – Zominy Laurent – Rossi Ombeline – Qian Xueyong**

## Table des matières


I. Architecture système : ..... 3

II. Rappel diagramme de la base de données : ..... 3

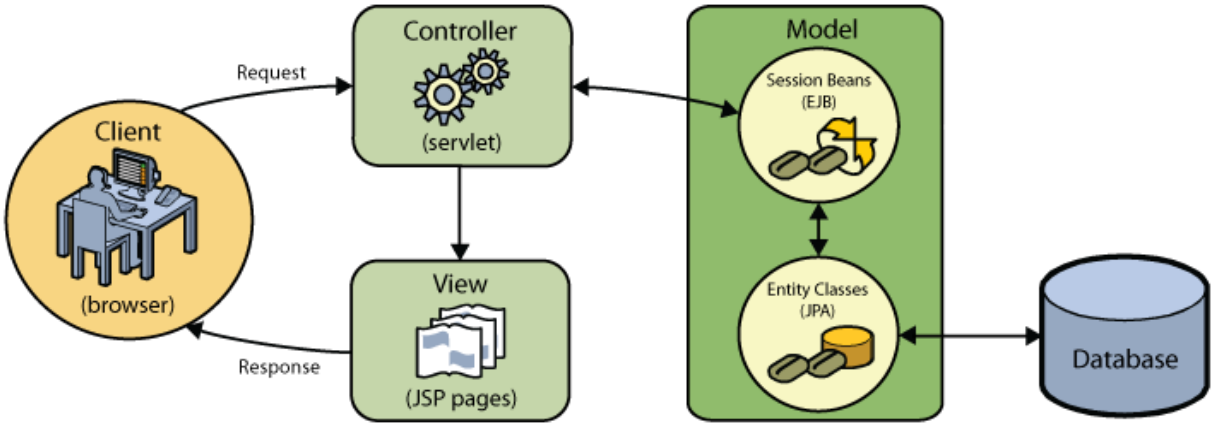
III. Les beans ..... 4

A. Les Entités : ..... 4

B. Les sessions ..... 6

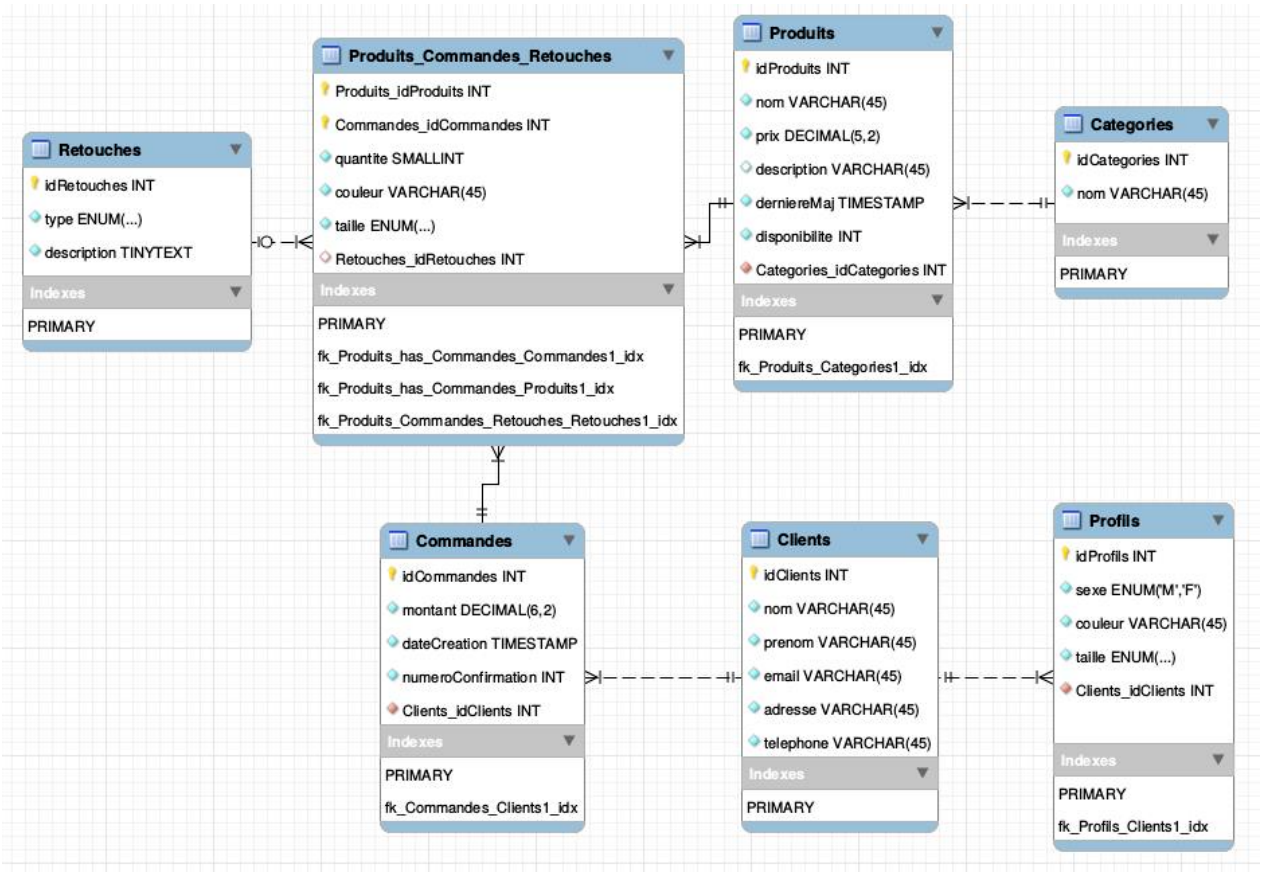
IV. Le controller : ..... 7

# I. Architecture système :




Dans notre application, le controller est représenté par la classe ControllerServlet. Les vues sont représentées par les pages JSP (cart.jsp, catégory.jsp, checkout.jsp, confirmation.jsp et index.jsp). Le model est représenté par l'ensemble des beans que nous allons détailler ci-après. (entity et session). Enfin, nous avons une base de donnée sous mySQL.

# II. Rappel diagramme de la base de données :



## III. Les beans

### A. Les Entités :

Pour développer notre version de notre site, nous avons mis en place divers composants.

Tout d'abord voici le contenu des entités qui compose notre application : Clients, Produits, Catégories, Commandes, Retouches et Profils.

Chaque entité correspond à une table dans notre base de données, et nous verrons par la suite les liens entre chacune de ces entités.

#### Clients :

- Attributs :
  - -id
  - -nom : nom de famille du client
  - -prénom : prénom du client
  - -email : email du client
  - -adresse : adresse de réception du client
  - -téléphone : téléphone du client
  - -cc_number : numéro carte de crédit du client

#### Produits :

- Attributs :
  - -id
  - -nom : nom du produit
  - -prix : prix du produit à l'unité
  - -description : description du produit
  - -disponibilité : reste en stock du produit
  - -date : date de dernière mise a jour du produit
  - -idCatégorie : clé qui fait référence a la table catégorie

#### Catégories :

- Attributs :
  - -id
  - -nom : nom de la catégorie

#### Commandes :

- Attributs :
  - -id
  - -montant : montant total de la commande
  - -dateCreation : date de création de la commande

-numeroConfirmation : numéro de confirmation de la commande

-idClients : id du client ayant crée la commande

## Retouches :

- Attributs :

-id

-type : type de retouche (« ourlet », « cintrage » etc…)

-description : description de la retouche (dimensions, emplacements etc …)

## Profils :

- Attributs :

-id

-sexe : masculin ou féminin

-couleur : couleur préférée

-taille : ses mensurations

-idclient : client ayant créé profil

## Produits_Commandes_Retouches :

- Attributs :

-idProduit : id du produit ayant été commandé

-idCOmmande : id de la commande a laquelle appartient le produit

-idRetouche :id de la retouche qui a été effectuée sur le produit (peut être null)

-quantité : quantité du produit commandé

-couleur : couleur du produit

-taille : taille du produit

Chaque entité est associée à une Bean « stateless » correspondant à une DAO (Data Access Object). Chaque DAO possède des méthodes permettant de communiquer avec la base de données (persist, merge, remove). De plus, chaque entité possède des fonctions qui leurs sont propres, généré automatiquement.

```
@NamedQueries({
    @NamedQuery(name = "Clients.findAll", query = "SELECT c FROM Clients c"),
    @NamedQuery(name = "Clients.findByIdClients", query = "SELECT c FROM Clients c WHERE c.idClients = :idClients"),
    @NamedQuery(name = "Clients.findByNom", query = "SELECT c FROM Clients c WHERE c.nom = :nom"),
    @NamedQuery(name = "Clients.findByPrenom", query = "SELECT c FROM Clients c WHERE c.prenom = :prenom"),
    @NamedQuery(name = "Clients.findByEmail", query = "SELECT c FROM Clients c WHERE c.email = :email"),
    @NamedQuery(name = "Clients.findByAdresse", query = "SELECT c FROM Clients c WHERE c.adresse = :adresse"),
    @NamedQuery(name = "Clients.findByTelephone", query = "SELECT c FROM Clients c WHERE c.telephone = :telephone"),
    @NamedQuery(name = "Clients.findByCcNumber", query = "SELECT c FROM Clients c WHERE c.ccNumber = :ccNumber")})
```

### ➢ Relation OneToMany :

Un produit possédé une seule et unique catégorie (jeans, robe etc…) et une catégorie peut être associée a plusieurs produits.
Un client possède 4 profils au maximum et un profil est créé par un seul et unique client. Un client peut effectuer un nombre indéterminé de commandes mais chaque commande est propre à un client

### ➢ Relation ManyToMany :

Le composant Produits_Commandes_Retouches est une association entre les 3 composants de même noms et permet de sauvegarder pour chacune des commandes la liste des produits achetés et les retouches dans le cas où il y en a.

## B. Les sessions

OrderManager : Cette classe java permet de gérer une commande. On utilise divers fonction pour ajouter un client dans la base, ajouter une commande, des produits, les retouches (s'il y en a) et remplir la table d'association entre ces 3 dernières.
Exemple pour la fonction avec le client :

```
// Add business logic below. (Right-click in editor and choose
// "Insert Code > Add Business Method")
private Clients addClient(String firstName, String familyName, String email, String phone, Strir
    Clients client = new Clients();
    client.setPrenom(firstName);
    client.setNom(familyName);
    client.setEmail(email);
    client.setTelephone(phone);
    client.setAdresse(address);
    client.setCcNumber(ccNumber);

    em.persist(client);
    return client;
}
```

Ces fonctions sont utilisées dans la fonction PlaceOrder qui permet d'assurer que le caractère transactionnel des ajouts dans la base (rollback en cas d'echec).

```java
@TransactionAttribute(TransactionAttributeType.REQUIRED)
public int placeOrder(String firstName, String familyName, String email, String phone, String ac
    try {
        Clients client = addClient(firstName, familyName, email, phone, address, ccNumber);
        Commandes order = addOrder(client, cart);
        Retouches retouches = addRetouches();
        addOrderedItems(order, retouches, cart);
        return order.getIdCommandes();
    } catch (Exception e) {
        @SuppressWarnings("ThrowableResultIgnored")
        Exception cause = (Exception) e.getCause();
        if (cause instanceof ConstraintViolationException) {
            @SuppressWarnings("ThrowableResultIgnored")
            ConstraintViolationException cve = (ConstraintViolationException) e.getCause();
            for (Iterator<ConstraintViolation<?>> it = cve.getConstraintViolations().iterator();
                ConstraintViolation<? extends Object> v = it.next();
                System.err.println(v);
                System.err.println("==>>" + v.getMessage());
            }
        }

        context.setRollbackOnly();
        return 0;
    }
```

De plus, chaque composant possède une sessionFacade permettant de faire le lien avec les entity. Enfin, nous possédons aussi un bean permettant l'envoi de mail pour confirmer la commande (EmailSessionBean)

## IV. Le controller :

Le controller « ControllerServlet ») permet de faire le lien entre les vues et les models.

```java
// if category page is requested
if (userPath.equals("/category")) {
    // get categoryId from request
    String categoryId = request.getQueryString();
    if (categoryId != null) {
        // get selected category
        selectedCategory = categoryFacade.find(Integer.parseInt(categoryId));
        // place selected category in request scope
        session.setAttribute("selectedCategory", selectedCategory);
        // get all products for selected category
        categoryProducts = selectedCategory.getProduitsCollection();
        // place category products in request scope
        session.setAttribute("categoryProducts", categoryProducts);
    }
```

On peut voir sur cette capture (ci-dessus) que le controller fait appel à l'entity bean (modèle correspondant à la Catégories).

```
// use RequestDispatcher to forward request internally
String url = "/WEB-INF/view" + userPath + ".jsp";

try {
    request.getRequestDispatcher(url).forward(request, response);
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Puis le controller fait appel à la vue Catégorie correspondant (la page jsp)