

Projet Darties – Groupe2

Dossier Architecture

**Création du document : Emric FORGE**

**Date : 14/12/2010**

Dossier Architecture

Projet Darties – Groupe2

# Contenu

[I. Contenu 1](#_Toc280112102)

[I. Introduction 2](#_Toc280112103)

[II. Respect du modèle MVC 2](#_Toc280112104)

[III. Utilisation d’un ORM (object-relational mapping) 3](#_Toc280112105)

[IV. Utilisation de Criteria et HQL 4](#_Toc280112106)

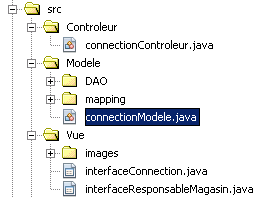
# Introduction

Ce dossier a pour but de rassembler l’ensemble des choix de fonctionnement pour développer l’application.

Les patterns, technologies, Framework utilisés seront décrits afin que les personnes intervenant sur le projet comprenne le développement de l’application rapidement.

# Respect du modèle MVC

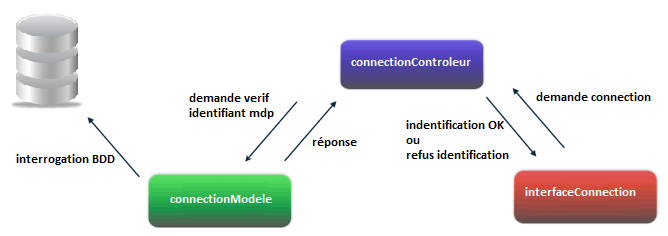
L’application sera développé selon le pattern MVC, afin d’avoir une séparation nette entre le traitement et la vue.



Le lien entre la vue et le modèle passe obligatoirement par le contrôleur. Par exemple pour la connexion :

Le fichier connectionControleur.java va instancier un connectionModele.java et un interfaceConnection.java

Lorsque l’utilisateur a saisi son identifiant et mot de passe, l’interface envoie les informations au contrôleur qui demande au modèle de vérifier l’authentification. Le modèle répond au contrôleur qui va agir sur la vue (refus ou acceptation connexion).



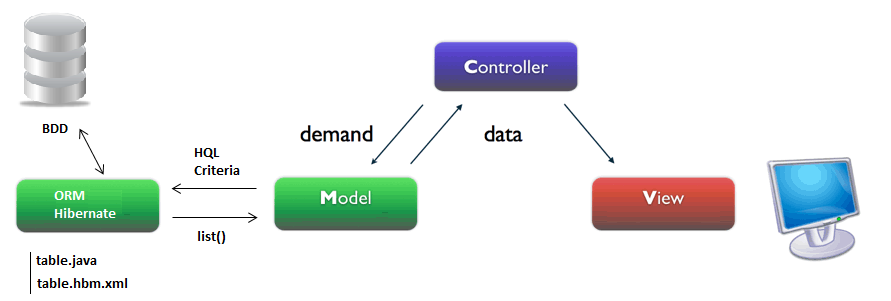
# Utilisation d’un ORM (object-relational mapping)

L’ORM mit en place est Hibernate, il va permettre de gérer la persistance des objets.

L’ensemble des tables de la BDD sont « mapper » au niveau de l’application. Chaque table est décrite grâce à un fichier .hbm.xml. Le fichier de mapping indique à Hibernate à quelle table dans la base de données il doit accéder, et quelles colonnes de cette table il devra utiliser.

Hibernate possède aussi un fichier de configuration hibernate.cfg.xml. Dans ce fichier est décrit la connexion à la BDD ainsi que les autres paramètres pour le bon fonctionnement d’Hibernate

Le mapping de la BDD va nous permettre de ne pas gérer l’accès aux données. Pour récupérer les données on passera par hibernate, pas de gestion de connexion…



Exemple d’utilisation de Hibernate avec le langage criteria

***public UtilisateurDAO() {***

***this.sessionfact = HibernateUtil.getSessionFactory();***

***}***

***public Utilisateur connection(String identifiant, String mdp) {***

***Utilisateur retour = null;***

***Session session = sessionfact.openSession();***

***session.beginTransaction();***

***Criteria crit = session.createCriteria(Utilisateur.class);***

***crit.setMaxResults(1);***

***crit.add(Restrictions.eq("identifiantutilisateur", identifiant));***

***crit.add(Restrictions.eq("mdputilisateur", mdp));***

***List<Utilisateur> utilisateurs = crit.list();***

***if(utilisateurs.size() == 1){***

***retour = utilisateurs.get(0);***

***}***

***session.getTransaction().commit();***

***session.close();***

***return retour;***

***}***

# Utilisation de Criteria et HQL

Criteria est une API de recherche orientée objet. Elle permet de faire des recherches dans la base de données de manière très simple. On peut ajouter des tris, des filtres, ...

Criteria est un langage pour hibernate permettant d’interroger facilement la BDD.

HQL est un langage d'interrogation extrêmement puissant qui ressemble au SQL. HQL est totalement orienté objet. Ce langage sera utilisé pour traiter des requêtes plus complexes cernant des notions comme l'héritage, le polymorphisme et les associations.