Plast’Prod

Projet Komodo

Spécifications techniques livrable 1 & 2 Version 0.3

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Modifié | Par | Action |
| 0.1 | 18/09/2012 | Cora Cox | Initialisation |
| 0.2 | 19/09/2012 | Sébastien Lavie | Livrable 1 & 2 |
| 0.3 | 08/11/2012 | Sylvain Zyssman | Correction typos |

[MCD et MOO 4](#__RefHeading__433_1897894644)

[1.MODELE CONCEPTUEL DE DONNEES 5](#__RefHeading__435_1897894644)

[2.MODELE ORIENTE OBJET 6](#__RefHeading__437_1897894644)

[Architecture technique 7](#__RefHeading__439_1897894644)

[3.Architecture applicative 8](#__RefHeading__441_1897894644)

[4.Architecture logicielle livrable 1 (E.R.P) 11](#__RefHeading__443_1897894644)

[5.Architecture logicielle livrable 2 (C.R.M) 12](#__RefHeading__445_1897894644)

[6.Architecture matérielle Livrable 1 (E.R.P) 13](#__RefHeading__447_1897894644)

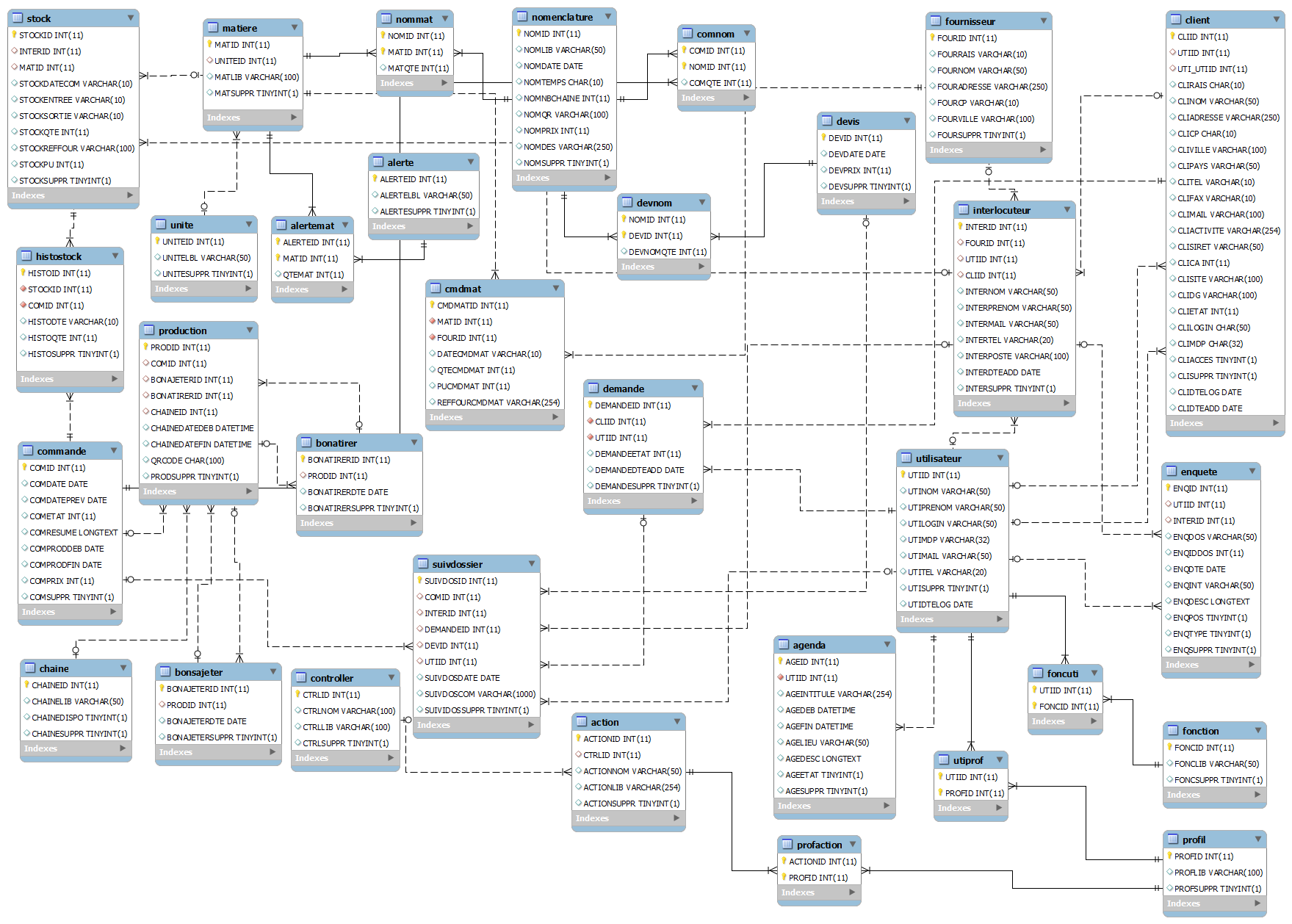
[7.Architecture matérielle Livrable 2 (C.R.M) 14](#__RefHeading__449_1897894644)

[8.Environnement de développement 15](#__RefHeading__451_1897894644)

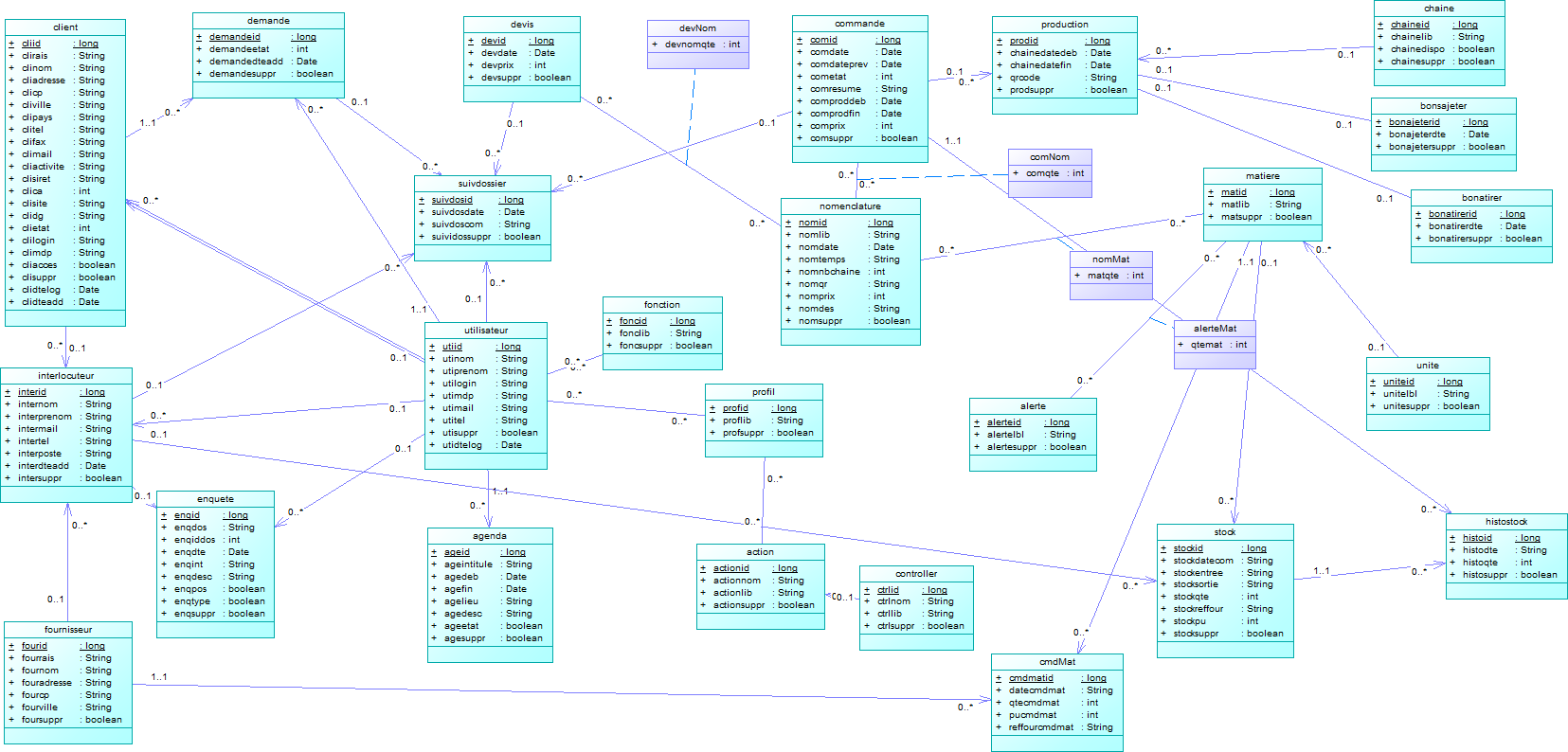
[Annexes 16](#__RefHeading__453_1897894644)

# MCD et MOO

## MODELE CONCEPTUEL DE DONNEES



## MODELE ORIENTE OBJET



# Architecture technique

## Architecture applicative

**Livrable 1 (E.R.P) & 2 (C.R.M) :**

L'application est réalisée en utilisant le modèle **M.V.C (Modèle, Vue et Contrôleur)**.

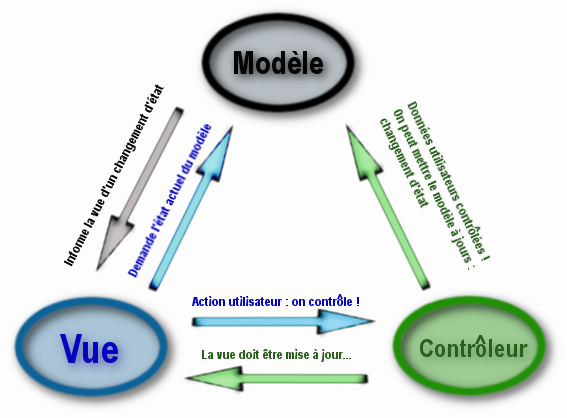
**Définition du modèle M.V.C :**

Modèle d'architecture qui cherche à séparer nettement les couches de présentation (UI : User Interface), métier (BLL : Business Logic Layer) et d'accès aux données (DAL : Data Access Layer).

Le but étant d'avoir une dépendance minimale entre les différentes couches de l'application. Ainsi les modifications effectuées sur n'importe quelle couche de l’application n'affectent pas les autres couches.

* Modèle : Encapsule le cœur fonctionnel de l'application, le domaine logique.
* Vue : Les données sont envoyées, par le modèle, à la vue qui les présente à l’utilisateur.
* Contrôleur : Reçoit les données et les transmets au modèle ou à la vue.

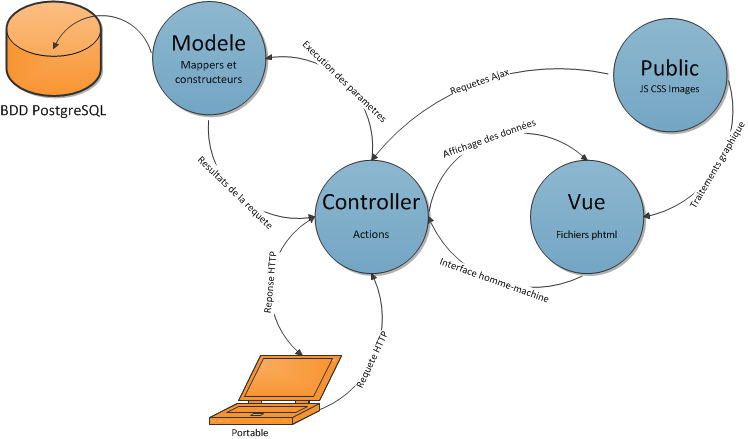
*Schéma graphique du modèle M.V.C :*



Mise en pratique du modèle M.V.C avec le livrable 1 (E.R.P) :

* Le client fait une demande au contrôleur (actions). Ce contrôleur voit passer toutes les demandes des clients. C'est la porte d'entrée de l'application. C'est le C de MVC.
* le contrôleur traite cette demande. Pour ce faire, il peut avoir besoin de l'aide de la couche métier (mapper pour la liaison à la base de données et ACL pour la gestion des droits), ce qu'on appelle le modèle M dans la structure MVC.
* le contrôleur reçoit une réponse de la couche métier. La demande du client a été traitée. Celle-ci peut appeler plusieurs réponses possibles. Un exemple classique est :
  + une page d'erreurs si la demande n'a pu être traitée correctement
  + une page de confirmation
* le contrôleur choisit la réponse (= vue fichier .phtml) à envoyer au client. Celle-ci est le plus souvent une page contenant des éléments dynamiques. Le contrôleur fournit ceux-ci à la vue.
* la vue est envoyée au client. C'est le V de MVC.

*Représentation graphique du modèle MVC avec le Framework Zend :*



Mise en pratique du modèle M.V.C avec le livrable 2 (C.R.M) :

* Le client fait une demande au contrôleur. Ce contrôleur voit passer toutes les demandes des clients. C'est la porte d'entrée de l'application. C'est le C de MVC.
* le contrôleur traite cette demande. Pour ce faire, il peut avoir besoin de l'aide de la couche métier (mapper pour la liaison à la base de données), ce qu'on appelle le modèle M dans la structure MVC.
* le contrôleur reçoit une réponse de la couche métier. La demande du client a été traitée. Celle-ci peut appeler plusieurs réponses possibles. Un exemple classique est :
  + une page d'erreurs si la demande n'a pu être traitée correctement
  + une page de confirmation
* le contrôleur choisit la réponse (= vue) à envoyer au client. Celle-ci est le plus souvent une page contenant des éléments dynamiques. Le contrôleur fournit ceux-ci à la vue.
* la vue est envoyée au client. C'est le V de MVC.

## Architecture logicielle livrable 1 (E.R.P)

### Briques utilisÉes

#### Modules standards

* PDO PostgreSQL pour la connexion à la base de données

#### Modules développés sur mesure

* ACL (Access Control List) pour gérer les accès des utilisateurs
* Module "HELP" (php & javascript)

#### Framework additionnel (GUI, effets graphiques…)

* Zend Framework (PHP)
* Zend PDF (PHP)
* PHP QR Code (PHP)
* Table Filter (JavaScript)
* EasyPaginate (JavaScript)
* JQuery (JavaScript)

### Plateforme

#### Base de données

* Le SGBD utilisé est **PostgreSQL**

#### Serveur http

* Le serveur web utilisé est **Wamp** avec le serveur http **Apache2**

### OS

* L'application est compatible avec les O.S (Windows, Mac et Linux) et testée sur 4 navigateurs (Chrome, Opéra, Firefox et IE8)
* La partie serveur Web sera installé sur un O.S linux (A préciser lors du livrable 3)

## Architecture logicielle livrable 2 (C.R.M)

### Briques utilisÉes

#### Modules développés sur mesure

* Table dispatcher qui va permettre d'envoyer des tableaux de données venant de la base de données
* Module de réplication pour récupérer la base en ligne sur le poste client (Les softs pgdump, pgrestore, createdb et dropdb sont utilisés pour répliquer la base de données )

#### Framework additionnel (GUI, effets graphiques…)

* Hibernate pour la connexion à la base de données et la gestion du mode connecté déconnecté

### Plateforme

#### Plateforme dédiée à l’intégration continue

* Le serveur d'application JBoss sera mis en place dans l'environnement de développement

#### Base de données

* PostgreSQL

#### Serveur http

* Apache2

### OS

* L'application est compatible avec les O.S (Windows, Mac et Linux)
* L'installation d'une J.V.M et de PostgreSql est nécessaire pour son fonctionnement.

## Architecture matérielle Livrable 1 (E.R.P)

### MatÉriel

## Architecture matérielle Livrable 2 (C.R.M)

### MatÉriel

* Le matériel est fourni par l'entreprise. Des ordinateurs portables sont à prévoir pour l'utilisation du livrable 2 (C.R.M)

## Environnement de développement

### Convention de nommage

#### Nommage des programmes

* Les applications sont nommées par les livrables (L1 & L2)

#### Nommage des fichiers

* Les fichiers sont nommés par leur fonctionnalité, placés dans un dossier nommé par le module puis placés dans l'arborescence MVC

### Gestion des versions

#### Livrable 1 -- V1 :

* Gestion des versions avec SVN installé sur un serveur virtualisé qui a pour O.S Debian.

#### Livrable 2 et Livrable 2 --- V2 :

* Gestion des versions avec GitHub en ligne.

# Annexes