# Architecture réalisée

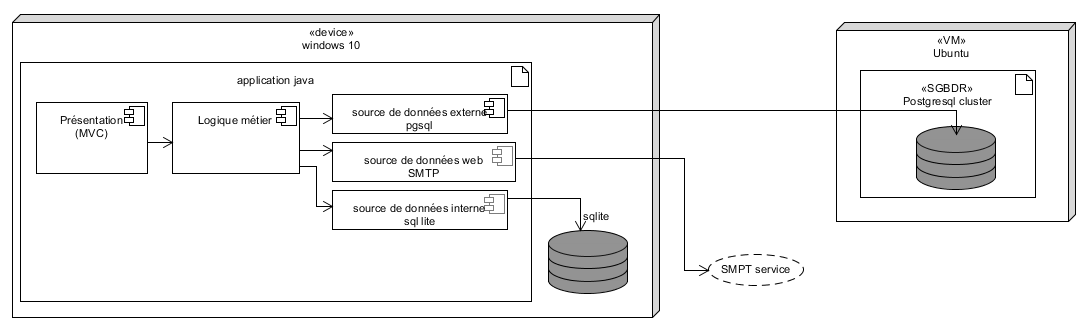
Voici l’illustration de l’architecture réalisées.

Figure Shéma général de l'architecture

Pour le projet money thoring nous avons mis en place l’architecture illustrée par la figure 1 : Shéma général de l’architecture. L’architecture est divisée en deux tiers :

1. Le serveur de base de données («VM» Ubuntu) avec PostgerSQL
2. Les clients («device» Windows 10) sur lesquels l’application java est déployées

Le premier tiers est le serveur de base de données PostgreSQL qui fait office de base de données centralisée et s’utilise quand les utilisateurs travail directement en ligne. Ce serveur est une machine virtuelle ubuntu. Les deuxièmes tiers sont les applications clientes sur chacun de postes client windows 10.

L’application cliente est principalement décomposée en trois couches :

1. La chouche présentations
2. La couche logique métier
3. Les couches sources de données

Dans le projet ces trois couches sont placé dans trois packages différents qui sont respectivement appelé :

1. gui (Graphique User Interface) qui gère les interaction et la présentation à l’aide d’interfaces graphiques pour les utilisateurs
2. bll (Business Logic Layer) qui gère la logique applicative.
3. dal (Data Access Layer) gère la persistances des données.

## Couche présentation

## Couche logique métier

## Couches d’accès aux données

La couche d’accès aux données encapsule les mécanismes propres à la persistances des données, elle permet de manipuler aisément les données stockées dans PostgreSQL et Derby. Voici la structure des packages de la couche d’accès aux données.



Figure Structure de la couche d'accès au données

**dal.exception** regroupe les exceptions que la couche d’accès aux données peut renvoyer aux couches appelantes. Ce package permet également de traiter les exceptions ou d’y inclure des classes de gestion d’exception.

**dal.ientities** Ce package englobe toutes les interfaces du modèle objet d’accès au données, ces interfaces sont implémentées dans les packages dal.entities.pgsql et dal.entities.derby

**dal.entities.derby** Ce package contiens l’implémentations des interfaces dal.ientities pour le mapping objet derby, le modèle objet derby est tagué avec les annotations d’Hibernate afin de mapper le modèle objet derby au modèle de base de données derby. Donc à chaque implémentation correspond une équivalence sous forme de tables en base de donnée selon la structure, la configuration du fichier de mapping Hibernate et des annotations.

**dal.entities.derby** Idem que pour derby, ce package contiens l’implémentations des interfaces dal.ientities pour le mapping objet PostgreSQL, le modèle objet PostgreSQL est tagué avec les annotations d’Hibernate afin de mapper le modèle objet PostgreSQL au modèle de base de données PostgreSQL. Donc à chaque implémentation correspond une équivalence sous forme de tables en base de donnée selon la structure, la configuration du fichier de mapping Hibernate et des annotations.

**dal.irepository** Ce package contiens les interfaces qui permettent d’effectuer des opérations en base de donnée tel que : Création, modification, suppression ainsi que d’autre traitements comme récupérer toutes les entités d’une association. Les opérations public dans les interfaces de ce package s’applique aussi bien pour Derby que pour pour PostgreSQL, c’est pour ces raison que ces interfaces n’utilise aucune implémentation mais uniquement les interfaces contenue dans dal.ientities.

**dal.repository.derby** Ce package regroupe les classes qui implémentes les interfaces du package dal.irepository. Dans ce package l’implémentation est uniquement réservée à derby.

**dal.repository.pgsql** Comme pour les repository derby, ce package regroupe les classes qui implémentes les interfaces du package dal.irepository. Il est uniquement question de l’implémentation propre à postgresql.

**dal.orm** A défaut d’avoir trouvé un nom convenable pour les classes quLe package dal.orm contiens l’interfaces (IORM) et les classes