Secteur Tertiaire Informatique
Filière étude - développement.

Développer une application n-tier

Positionner les composants de la couche présentation dans l'architecture x-tiers

Positionner la couche présentation dans une architecture JEE

Accueil

Apprentissage

Période en entreprise

Evaluation

SOMMAIRE

I ACRONYMES UTILISES

IHM Interface Homme Machine
JSP Java Server Page
MVC Modèle Vue Contrôleur

II OBJECTIFS

L'objectif de ce document est de situer la couche PRESENTATION dans une architecture multi couches, ainsi que son rôle et ses interactions avec les autres couches de l'application informatique.

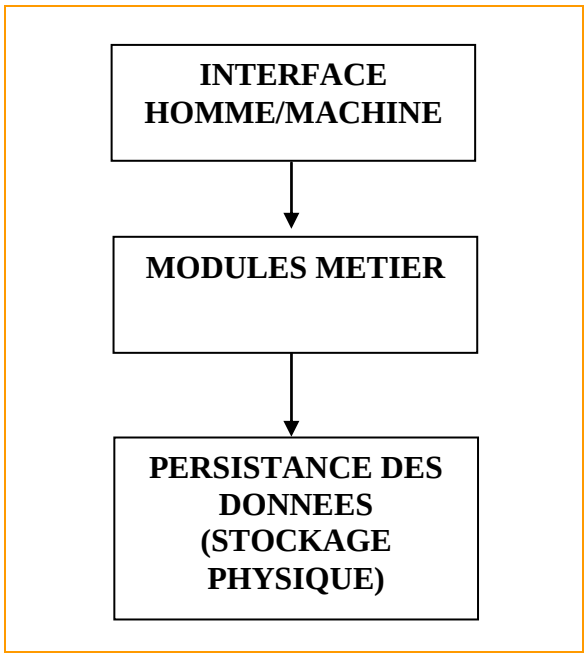
III ARCHITECTURE EN COUCHES

III.1 PRINCIPAUX NIVEAUX D'UNE APPLICATION

Une application d'entreprise, au sens système d'informations, est un système logiciel que l'on peut décomposer « grossièrement » en trois niveaux modulaires qui correspondent à des niveaux de responsabilité :

- L'interface homme/machine (IHM) qui regroupe les interactions de l'utilisateur avec le système
- Un ensemble de modules métier qui implémente les responsabilités de la logique de l'application
- Un mécanisme de stockage externe qui assure la persistance physique des données du système.

Le graphe ci-dessous schématise ces niveaux modulaires.



On reconnaît dans ce graphe trois niveaux de responsabilités représentés par les rectangles. Les traits indiquent une association de dépendance et la flèche donne le sens du couplage.

Ainsi, l'Interface Homme/Machine utilise les services exposés par les modules métier qui, eux-mêmes, utilisent les services du mécanisme de stockage physique.

III.2 STRUCTURATION EN COUCHES LOGIQUES

Une couche logique est une partie de l'architecture d'une application qui remplit une fonction précise, comme par exemple l'ensemble de l'interface avec l'utilisateur. Elle possède donc ses propres fonctionnalités et programmes d'exécution. **Elle représente un regroupement logique de composants logiciels qui implémente ses fonctionnalités**.

Une couche n'a de pertinence que si :

- Elle offre une réelle valeur ajoutée par rapport à la couche du dessous.
- Elle offre des services supplémentaires par rapport à ce que sait déjà faire la couche du dessus.

Le couplage entre les couches doit être faible alors que le couplage entre les composants à l'intérieur d'une couche doit être fort : ce principe permet de tester chaque couche de manière indépendante.

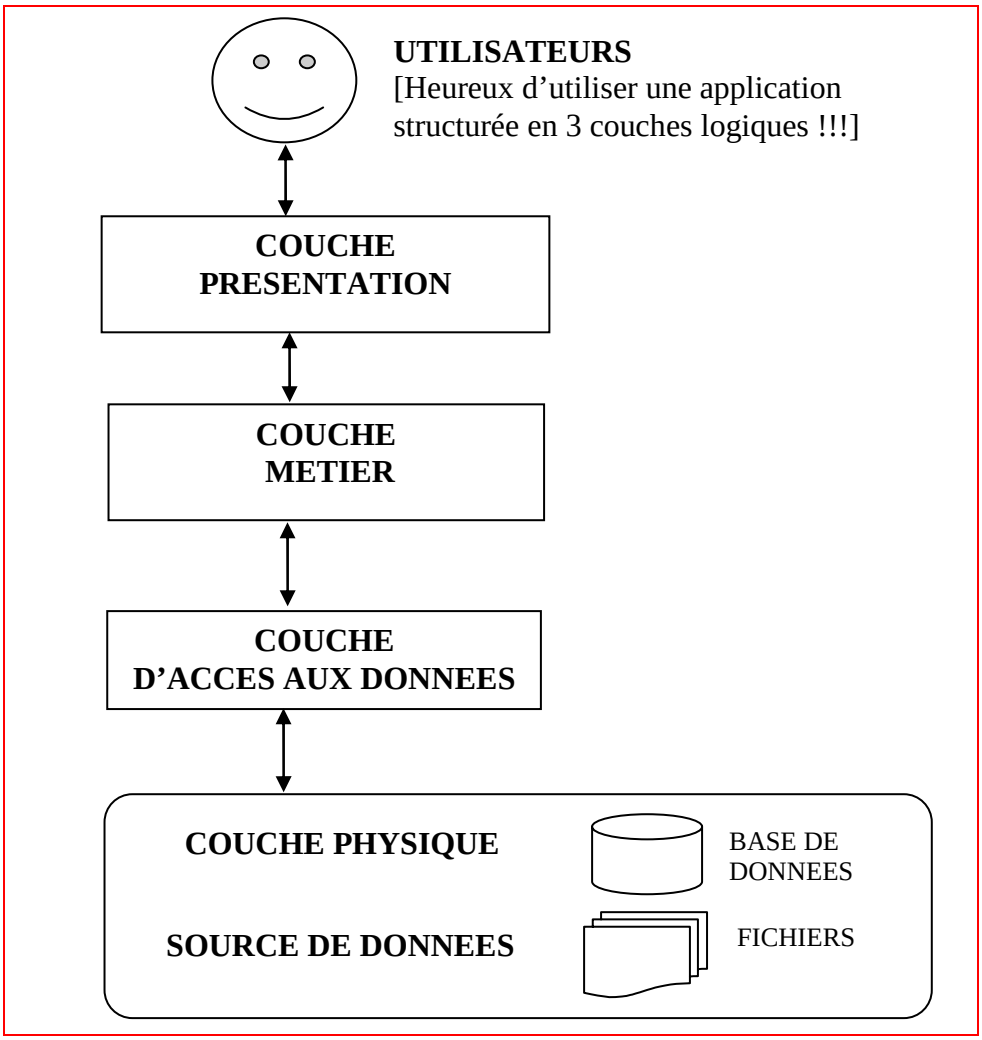
A partir des trois niveaux décrits dans le paragraphe précédent, nous pouvons définir une architecture classique en 3 couches logiques (layers) de notre application informatique.

- En premier lieu, on trouve la **couche PRESENTATION** (Presentation Layer). Elle correspond à la partie de l'application visible et avec laquelle les utilisateurs vont interagir. Elle se présente sous la forme de clients - on parle d'interface Homme-Machine - qui permettent de transcrire à l'application les actions que veut mener un utilisateur.
- En second lieu, il y a la **couche METIER** (Business Layer). Elle s'interface entre la couche PRESENTATION et la couche d'ACCES AUX DONNEES. Elle a pour but d'effectuer le traitement métier. Elle est en général constituée d'une couche SERVICES et d'une couche OBJETS METIERS. Elle représente ainsi les objets métiers jugés persistants dans le diagramme UML et implémente leurs règles de gestion.
- Enfin, il y a la **couche d'ACCES AUX DONNEES** (Data Access Layer). Cette couche est chargée de la création, modification, récupération et suppression des données.

Nous pouvons éventuellement définir une 4ème couche.

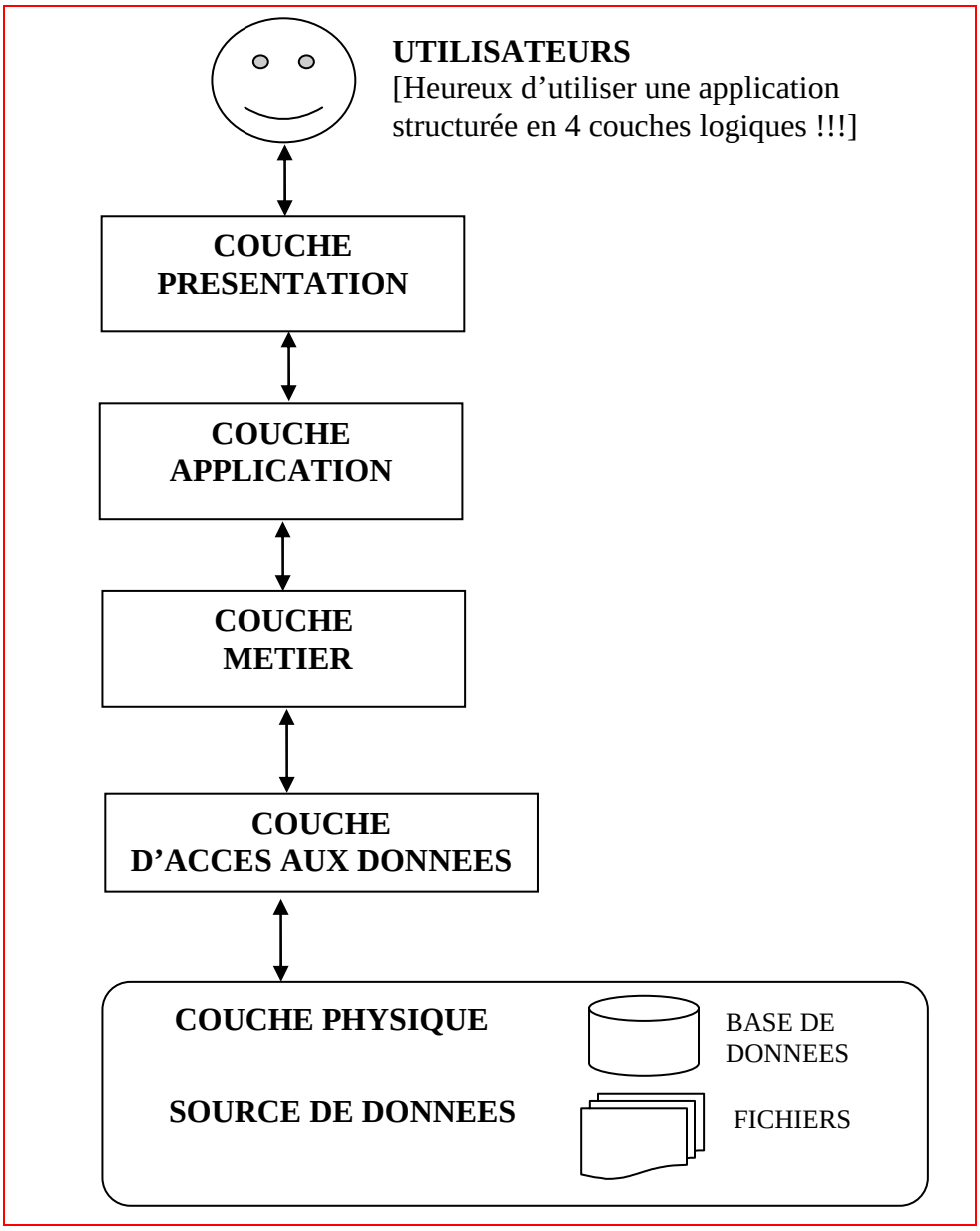
- La **couche PHYSIQUE** ou **GESTION DES DONNEES** ou **RESSOURCES** qui correspond à la structure physique des données : SGBD, annuaire LDAP, fichiers, …

Le graphe ci-dessous schématise ces trois couches logiques.



Dans certaines architectures, les responsabilités de la couche PRESENTATION sont séparées en 2 couches distinctes. Ainsi, une couche APPLICATION est créée entre la couche PRESENTATION et la couche METIER.
Il est à noter que ces deux couches, PRESENTATION et APPLICATION, sont étroitement liées et sont souvent regroupées ou peuvent interagir en se reposant sur des frameworks dédiés.

Le graphe ci-dessous schématise ces quatre couches logiques.



Dans cette architecture, les responsabilités sont reparties de la façon suivante :

- **Couche PRESENTATION** : Cette couche représente uniquement l'interface graphique utilisateur. C'est dans cette couche qu'intervient le formatage des affichages. Toute requête utilisateur (demande d'authentification, ajout/modification/suppression d'élément,…) effectuée ici est envoyée à la couche APPLICATION.
- **Couche APPLICATION** : Cette couche, reliée à la couche PRESENTATION, prend en charge la gestion des événements induits par la couche PRESENTATION (navigation entre écrans, clics, gestion des éléments graphiques…). Elle constitue le lien entre la couche PRESENTATION et la couche METIER, entre chaque action de l'utilisateur et son action associée définie dans la couche SERVICE de la couche METIER.

Un point important concerne la communication entre les couches. Une bonne pratique consiste à **faire communiquer les couches entre elles par l'intermédiaire d'interfaces** et de protocoles, de préférence standardisés.

Les services d'une couche sont mis à disposition de la couche supérieure. En se basant sur ce principe, on évite donc, lorsqu'on développe une application, qu'une couche invoque les services d'une couche plus basse que la couche immédiatement inférieure ou plus haute que la couche immédiatement supérieure. Chaque couche ne communique qu'avec ses voisines immédiates.

III.3 AVANTAGES D'UNE STRUCTURATION EN COUCHES

La structuration d'une application en couches logicielles ne provient pas d'un délire intellectuel dont l'objectif principal est de se faire plaisir.
L'architecture en couches permet de percevoir un certain nombre d'avantages que l'on peut résumer au travers des points suivants :

- Permettre un déploiement des composants en environnement distribué.
- Cadrer le travail des équipes de réalisation. Les développeurs peuvent se spécialiser dans un type de couche.
- Faciliter la maintenance de l'application. Une architecture banalisée de la sorte est plus facilement maintenable dans la mesure où un nouvel intervenant est capable à partir de la description de l'architecture de comprendre le rôle de chaque composant développé et de s'insérer rapidement dans cette logique.
- Concentrer la complexité de mapping entre l'application Orientée Objets et le modèle relationnel du SGBD dans une seule couche (accès aux données).
- Faciliter l'évolution de l'application :
  - Si vous décidez d'utiliser un autre type de gestionnaire de persistance de données qu'une base de données relationnelle (BD Objet, fichiers classiques, données XML, ...), seule la couche d'accès aux données est à modifier.
  - Si vous désirez disposer de plusieurs interfaces utilisateurs distinctes (WEB, PDA, etc.), vous vous contenterez d'écrire une autre couche de présentation en appelant la même couche METIER.

La mise en œuvre d'une telle architecture peut paraître lourde dans le cadre de petits projets informatiques. Pour des applications de plus grande envergure, il est illusoire de se passer de ce type d'architecture.

III.4 COUCHES ET TIERS

Dans la littérature, les termes tier et couche sont souvent utilisés indifféremment pour désigner la même chose. Il existe cependant une différence qui ne doit pas être négligée.

Les couches concernent la division logique des composants et des fonctionnalités de l'application informatique et ne prennent pas en compte la localisation physique des composants sur différents serveurs et dans différents lieux géographiques.

Les tiers concernent la distribution physique des composants et des fonctionnalités sur différents serveurs, ordinateurs, réseaux et localisations distantes.

Même si les couches et les tiers utilisent les mêmes termes (présentation, métier, service et données), souvenez-vous que seuls les tiers impliquent une séparation physique. A noter qu'il est assez commun de localiser plus d'une couche sur une même machine physique.

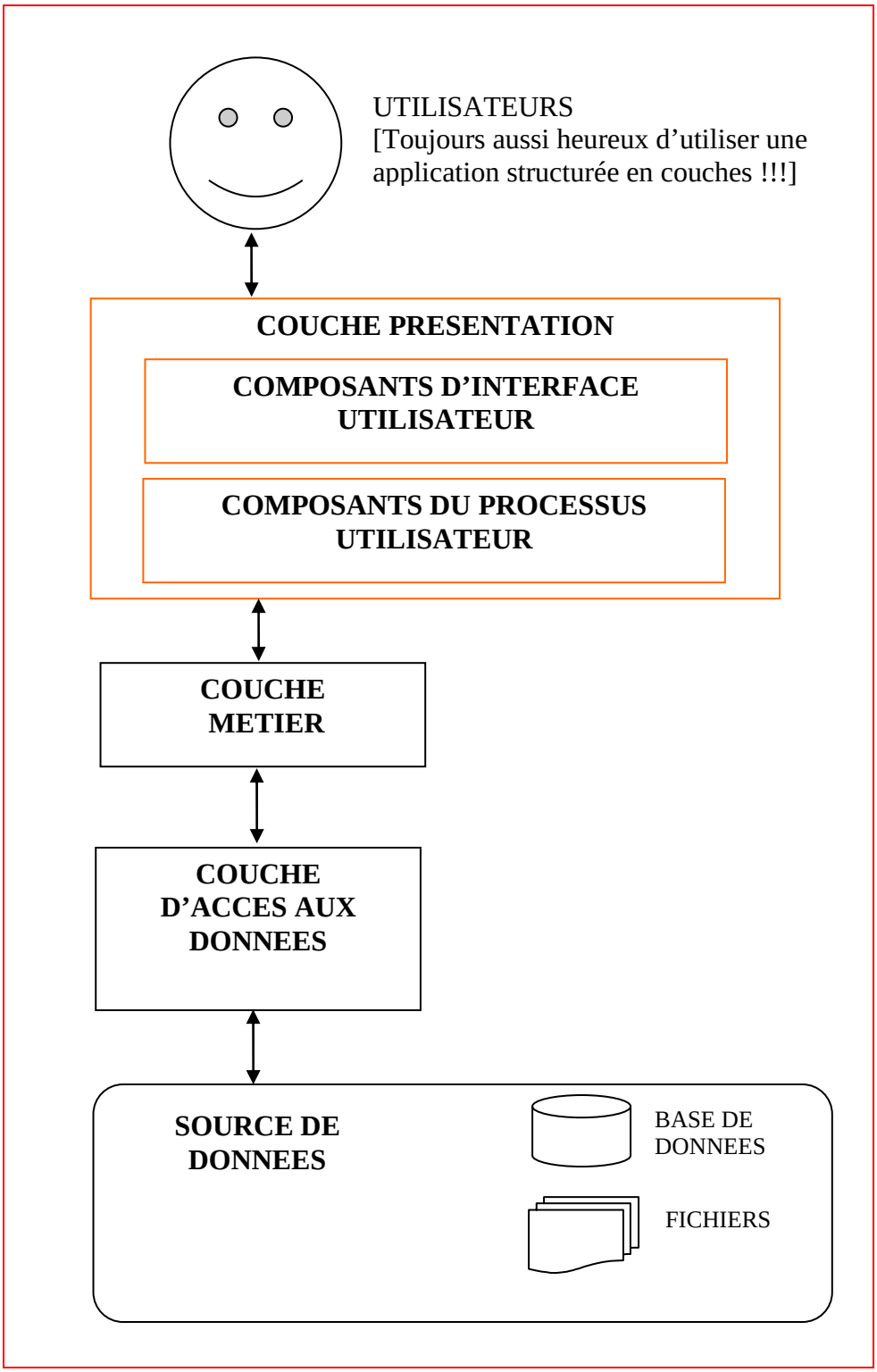
En résumé, nous pouvons dire que les couches sont déployées sur des tiers.

IV COUCHE PRESENTATION

IV.1 ROLE

La couche PRESENTATION est chargée de l'affichage des informations à l'utilisateur final. Cette couche contient deux types de composants :

1. **Composants d'interface utilisateur (User Interface components**), appelés aussi **composants de rendu** : ces composants sont chargés d'effectuer le rendu à l'utilisateur final (IHM). Ils mettent en forme les données pour l'affichage et ils acquièrent et valident les données saisies par les utilisateurs.
2. **Composants du processus utilisateur (User Interface process components)** appelés aussi **composants de gestion des interactions** : ces composants sont - chargés de la logique d'enchainement entre les écrans. Ils gèrent la synchronisation et l'orchestration des interactions avec les utilisateurs.



IV.2 MODELISATION DES CLASSES DE LA COUCHE PRESENTATION

IV.3 IDENTIFICATION DES CLASSES D'ANALYSE

Lors de la phase d'analyse, les classes d'analyse décrites ci-dessus peuvent être identifiées.

Classe <<boundary>> ou <<frontière>>

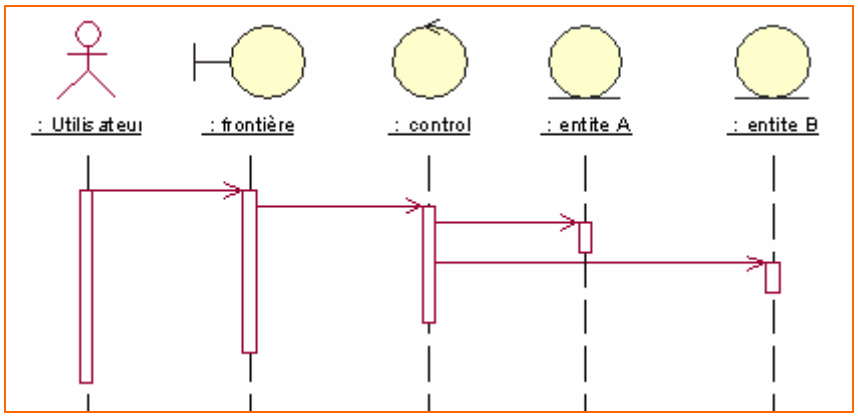
- On trouve ces classes :
  - Entre un acteur (humain) et le système (généralement graphique). La classe est alors une abstraction des écrans identifiés lors de l'étape d'identification des écrans
  - Entre le système et les acteurs système (Interface)
- Normalement il y a une classe <<boundary>> par couple Acteur/Cas d'utilisation
- Une classe <<boundary>> regroupe plusieurs classes écrans.
- Dans cette activité on ne s'occupe pas de l'enchaînement des écrans.
- On suppose que l'IHM permet à l'acteur de communiquer avec l'application.

Classe <<control>> ou <<contrôle>>

- Contrôle associé à un scénario d'un cas d'utilisation
- Au moins une classe <<control>> candidate par scénario, mais cette classe peut se retrouver dans plusieurs scénarii

Classe <<entity>> ou <<entité>>

- Ce sont les objets métier du système destinés à être sauvegardés.

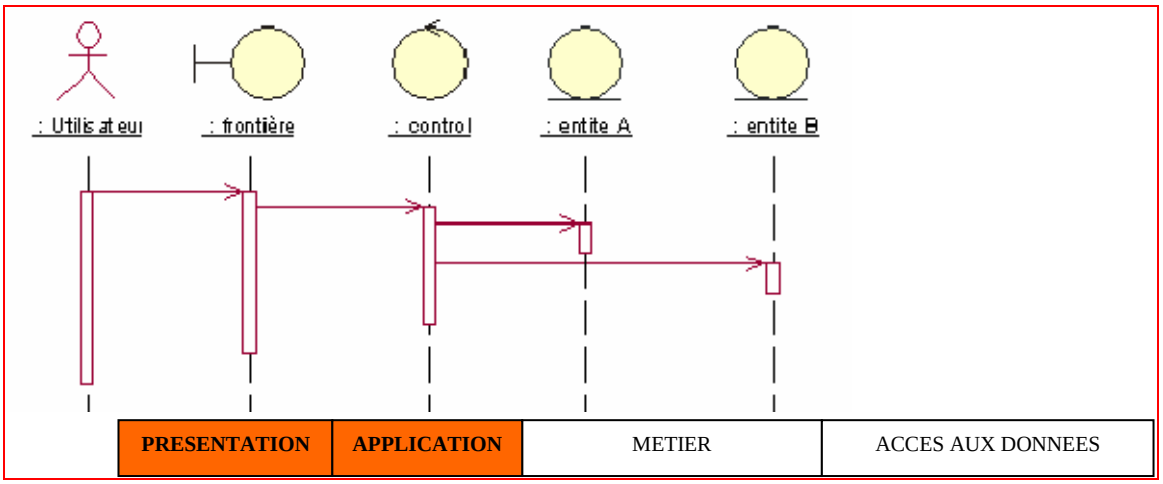


Note : vous pouvez utiliser les stéréotypes UML pour marquer les classes <<boundary>>, <<control>> et <<entity>>.

IV.4 REPARTITION DES CLASSES DANS LES COUCHES

Une fois les classes d'analyse identifiées, vous pouvez les répartir entre les différentes couches.

- Les classes << boundary >> dans la couche Présentation
- Les classes <<control>> dans la couche Application
- Les classes <<entity >> dans la couche Métier



IV.5 PRINCIPES ET BONNES PRATIQUES

Il existe plusieurs points importants que vous devez considérer quand vous concevez une couche PRESENTATION. L'utilisation de certains principes et le respect de certaines bonnes pratiques vous permettront de mettre toutes les chances de votre côté pour réaliser une conception en parfaite corrélation avec les exigences exprimées par le client.

Les étapes listées ci-dessous décrivent l'approche que vous pouvez adopter pour concevoir la couche PRESENTATION de votre application :

- **Identifier le type de client** : choisir un type de client (client léger, lourd ou riche) qui satisfasse les exigences du projet et adhère aux contraintes d'infrastructure et de déploiement de l'organisation cible. Les différents types de client sont abordés dans le document intitulé « ***Identifier les technologies de la couche Présentation*** ».
- **Déterminer comment vous présenterez les données** : choisir le format des données pour la couche PRESENTATION et décider comment vous présenterez les données au niveau de l'interface utilisateur.
- **Définir la stratégie de validation des données** : définir les techniques de validation des données pour protéger l'application d'entrées incorrectes.
- **Déterminer la stratégie adoptée pour la logique métier** : Définir la logique métier afin de la découpler au code de la couche PRESENTATION.
- **Déterminer votre stratégie pour la communication avec les autres couches** : si l'application doit être conçue en couches, avec une couche d'accès aux données et une couche métier, déterminer une stratégie de communication entre la couche PRESENTATION et les autres couches de l'application.
- **Appliquer les "design patterns"**. Mettre en place des solutions éprouvées comme par exemple le design pattern MVC.
- **Considérer les guides d'interface utilisateur**. Des constructeurs (Apple, SUN, Microsoft, …) proposent des « bibles » qui définissent l'ensemble des normes à respecter pour les applications d'une même famille (WEB, WAP, …).
- **Adhérer aux principes de conception orientée utilisateur**. Avant de concevoir votre couche PRESENTATION, il est nécessaire de comprendre les besoins de l'utilisateur. Par l'intermédiaire d'enquêtes, d'interviews, de définition de cas d'utilisation, vous pourrez déterminer la meilleure conception permettant de couvrir les besoins du client.

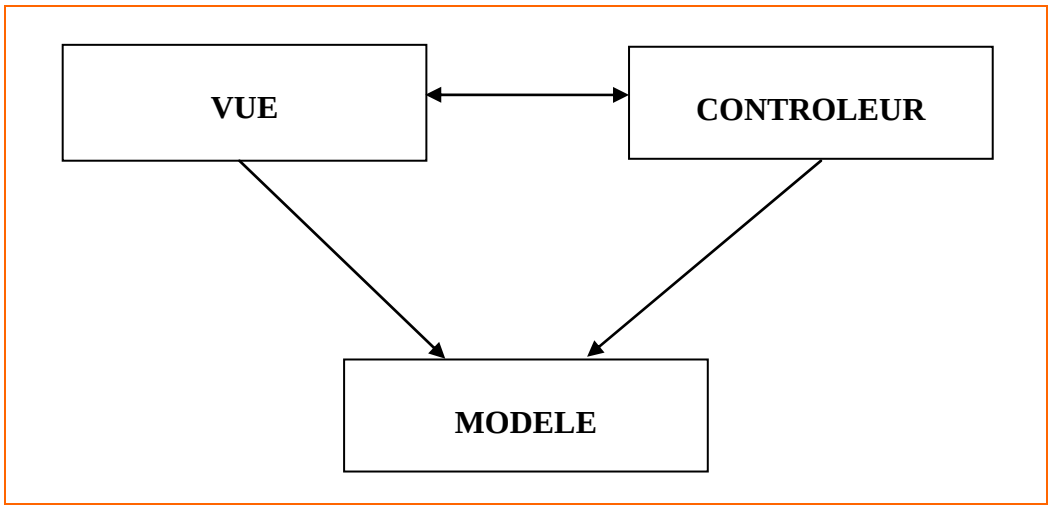
V COUCHE PRESENTATION ET LES DESIGN PATTERNS

Pour la conception et l'implémentation de la couche PRESENTATION, vous pouvez vous appuyer sur un design pattern architectural. Le plus connu et le plus utilisé à ce jour est le concept **Modèle Vue Contrôleur**.

V.1 CONCEPT MODELE – VUE - CONTROLEUR

Le concept **Modèle Vue Contrôleur ou MVC** a été introduit dans les années 80 dans le langage SmallTalk. L'idée est de séparer la couche modèle de la couche PRESENTATION, l'interaction entre ces deux couches se fait par l'intermédiaire d'un contrôleur. Ce modèle de conception permet de ne pas mélanger les différents aspects techniques de l'application afin de pouvoir facilement réutiliser le code.

Ainsi dans le modèle MVC toutes les données de l'entreprise ainsi que la logique métier associée peuvent être représentées par la partie Modèle. La Vue, quant à elle, accède aux données à travers le Modèle et décide comment les afficher au client. La Vue doit s'assurer que cette présentation change lorsque les données du modèle sont modifiées.
Le Contrôleur enfin, peut interagir avec la Vue et convertir les actions du client qui sont comprises et interprétées par la couche Modèle. Le Contrôleur décide également quelle est la prochaine vue à présenter au client en fonction de sa dernière action et des résultats des actions correspondantes au niveau du Modèle.



Le modèle MVC est recommandé par SUN Microsystems comme architecture des applications JEE.

Dans la littérature sur les technologies Web, on retrouve souvent au sein des plates formes JEE les termes de Modèle MVC1 et Modèle MVC2 sans pourtant en découvrir la signification. Cette terminologie issue des premières ébauches de la spécification de la technologie JSP décrit en fait deux modèles de conception basique de pages JSP. Alors que ces termes ont disparu du document de spécification de la technologie en question, ils ont tout de même survécu dans le langage commun des architectes. Le Modèle MVC1 et le Modèle MVC2 font simplement respectivement référence à l'absence ou au contraire à la présence d'une servlet contrôleur répondant aux requêtes variées du client utilisateur de l'application définie par ces modèles.

V.1.1 MODELE MVC1

L'architecture du Modèle MVC1 consiste en un simple navigateur Internet qui accède directement aux pages JSP. Les pages JSP accèdent aux classes Java JavaBeans du serveur d'applications qui représentent le modèle d'application. L'affichage de la page suivante est déterminé soit par les liens hypertextes présents dans le document source soit par les paramètres de la requête client. Le contrôle de l'application dans le cadre du Modèle 1 est décentralisé, de sorte que la page courante affichée dans le navigateur détermine la page qui suivra. De plus, chaque page JSP traite elle-même ses paramètres d'entrées.

V.1.2 MODELE MVC2

L'architecture du Modèle 2 permet d'introduire la notion de Servlet Contrôleur entre le navigateur et les pages JSP ou Servlets délivrées suite à la requête client. Le contrôleur a pour tâche de centraliser la logique de façon à rediriger les requêtes vers la vue suivante à afficher, cette dernière étant déterminée par l'URL, les paramètres d'entrée et l'état de l'application. Les applications basées sur le Modèle 2 sont plus faciles à maintenir et à faire évoluer car chacune des pages ne fait pas directement référence à une autre. De plus le Servlet contrôleur représente le seul point d'entrée de l'application et simplifie ainsi la gestion de la sécurité et le contrôle de la fréquentation.
Pour ces raisons, c'est ce modèle qui est recommandé la plupart du temps pour les applications interactives.

Oeuvre collective de l'AFPA

Etablissement référent
AFPA Toulouse Palays

Equipe de conception
Pascal DANGU

Date de mise à jour 2009
afpa © Date de dépôt légal mois année

afpa / Direction de l'Ingénierie13 place du Générale de Gaulle / 93108 Montreuil Cedex
association nationale pour la formation professionnelle des adultes
Ministère des Affaires sociales du Travail et de la Solidarité