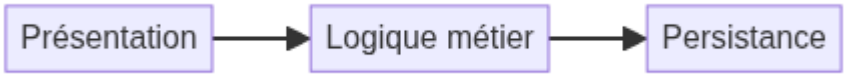
Purifions l’architecture 3 tiers

1. L’impureté de l’architecture 3 tiers

L’architecture 3 tiers présente l’inconvénient d’être impure : la dépendance qu’elle implique de sa couche logique métier vers sa couche persistance donne lieu à une contamination.



En effet, la couche persistance est par nature impure, puisque sa vocation même est de réaliser des opérations d’entrées et de sorties (en l’occurrence vers des supports persistants de stockage de données).

La couche logique métier, quant à elle, aurait l’opportunité d’être composée de code pur. Mais sa dépendance vers la couche persistance la rend impure par contamination.

2. Purifions tout cela

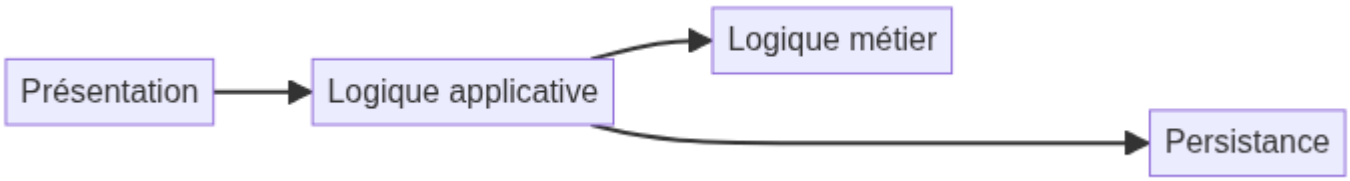
On peut tenter de purifier cette architecture en introduisant une **distinction entre logique applicative et logique métier**.

La **logique métier** demeure constituée des règles de gestion et traitements métier à appliquer aux données. Mais elle est désormais déchargée de la responsabilité d’avoir à demander par elle-même ces données auprès de la couche persistance.

Au lieu de cela, les données à traiter devront lui être fournies par les composants logiciels appelants, sur le modèles des arguments fournis aux fonctions pures en mathématiques et en programmation à la mode fonctionnelle.

Le fait d’aller chercher ces données quelque part, en d’adressant à la couche persistance, relève en fait de la **logique applicative**. Cette dernière définie la façon dont sont agencées et coordonnées les actions à mener en réponse aux demandes de l’utilisateur·ice.

Pilote de l’application, elle est consciente de son environnement technique, précisément pour que la logique métier n’aie pas à l’être.



On peut également imaginer une éventuelle dépendance de la couche persistance vers la couche métier, pour ce qui est de récupérer la définition des structures de données (entités) qui serviront de format de communication entre les couches logique applicative, logique métier et persistance. À défaut de mettre en place une telle dépendance, la couche logique applicative devra assurer des conversions entre les différentes structures de données utilisées par les différents composants, comme le fait la couche services dans le cadre d'une architecture contrôleurs / services / dépôts.

Mais si cette dépendance persistance → logique métier est mise en place, elle devrait se borner à la récupération des structures entités et éviter tout appel de règle de gestion. C'est le rôle exclusif de la couche logique applicative que de provoquer leur appel.

Le flux de traitements correspondant à une action de l'utilisateur·ice pourra prendre des formes légèrement différentes, dans l'ordre d'appel des différents composants logiciels, selon qu'il s'agit plutôt de collecter des données déjà connues pour les présenter à l'écran ou plutôt d'en ajouter dans le système (enregistrement de saisies).

Dans le cas d'une collecte de données, le flux des traitements prendra typiquement la forme suivante :

1. La couche présentation prend en compte l'action effectuée par l'utilisateur·ice.

   Cela se traduit par une demande qu'elle adresse à la couche logique applicative (et non plus à la couche logique métier). Cette demande peut s'accompagner de données (montantes), typiquement passées en paramètres aux opérations mises à disposition par la couche logique applicative. Ces données montantes seront notamment les critères de sélection (mots clés, dates définissant une période de publication…) correspondant aux données que l'utilisateur·ice souhaite collecter.

2. La couche logique applicative prend en compte cette demande, avec les données passées en paramètres.

   Elle identifie les traitements métiers qui devront être réalisés pour constituer la réponse à la demande et, avec eux, les données qui leur seront nécessaires.

3. Elle s'adresse alors à la couche persistance pour obtenir ces données, en lui passant des paramètres qui correspondent aux critères de sélection des données à récupérer.

4. La couche persistance effectue la collecte des données auprès des supports de stockage dont elle a la charge et les retourne à la couche logique applicative.

5. La couche logique applicative appelle ensuite la couche logique métier en lui passant les données qu'elle vient de recevoir de la couche persistance.

6. La couche logique métier prend en compte cette demande, avec les données passées en paramètres, et y applique ses règles de gestion pour élaborer sa réponse. C'est notamment le moment où les données stockées récupérées précédemment peuvent être complétées avec des données calculées (selon des règles métier).

   Les données ainsi filtrées et transformées sont donc retournées à la couche logique applicative.

7. La couche logique applicative va réceptionner ces données et les intégrer dans sa réponse à la couche présentation.

8. La couche présentation va enfin réceptionner les données retournées par la couche logique applicative, les mettre en forme et les afficher à l'utilisateur·ice.

Dans le cas de l'enregistrement d'une saisie de données, le flux de traitements sera plutôt le suivant :

1. La couche présentation prend en compte l'action effectuée par l'utilisateur·ice sur l'interface graphique.

   Cela se traduit par une demande qu'elle adresse à la couche logique applicative (et non plus à la couche logique métier). Cette demande peut s'accompagner de données (montantes), typiquement passées en paramètres aux opérations mises à disposition par la couche logique applicative. Ces données montantes seront principalement les données saisies que l'utilisateur·ice souhaite enregistrer.

2. La couche logique applicative prend en compte cette demande, avec les données passées en paramètres.

   Elle appelle la couche logique métier pour valider et éventuellement mettre en conformité ces données avant qu'elles soient enregistrées.

3. La couche logique métier réceptionne ces données et procède aux validations et éventuelles mises en conformité.

   Si la validation échoue, cela peut se traduire par une erreur qui permettra à la couche applicative de signaler à la couche présentation qu'il y a un problème de cohérence empêchant les données d'être enregistrées.

4. Si la validation réussit, la couche logique applicative passe à la couche persistance les données à enregistrer (éventuellement mises en conformité par la couche logique applicative).

5. La couche persistance procède à l'enregistrement et en informe la couche logique applicative.

   Si une erreur technique se produit (par exemple s'il n'y a plus d'espace de stockage disponible), elle en remonte l'information à la couche applicative.

6. La couche logique applicative informe la couche présentation du fait que l'enregistrement a été effectué ou lui remonte à son tour une éventuelle erreur survenue (soit une erreur fonctionnelle signalée par la couche logique métier, soit une erreur technique signalée par la couche persistance).

7. La couche présentation affiche pour l'utilisateur·ice un message de confirmation de l'enregistrement ou, le cas échéant, un message d'erreur.

La différence entre les deux scénarios consiste essentiellement dans l'ordre relatif dans lequel la couche logique métier d'une part et la couche persistance d'autre part seront appelées par la couche logique applicative. Dans le cas de la collecte de données c'est la couche persistance qui est appelée avant la couche logique métier alors que dans le cas de la saisie c'est la couche logique métier qui est appelée avant la couche persistance.

On peut encore imaginer un scénario de saisie dans lequel des données complémentaires doivent d'abord être collectées auprès de la couche persistance, pour ensuite être passées à la couche logique métier conjointement avec les données saisies pour validation et mise en conformité, avant que la couche persistance soit à nouveau appelée pour procéder à l'enregistrement effectif.

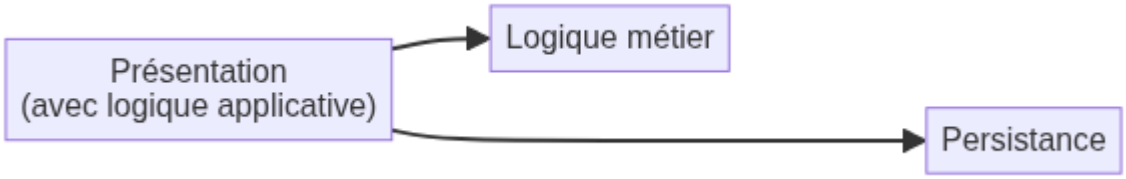
3. 3 ou 4 couches ?

Avec l'introduction d'une couche logique applicative à part entière, comme décrit précédemment, on passe de fait d'une architecture à 3 couches à une architecture à 4 couches.

Historiquement, l'un des enjeux de la mise en place de l'architecture 3 tiers était d'inciter les développeurs de l'époque, qui avaient éventuellement avant cela connu l'architecture client serveur, à mettre en place les traitements métier côté serveur plutôt que côté client.

Mais pour la logique applicative, la question peut se poser de la laisser côté client, si l'on considère que le programme client manipulé l'utilisateur.ice est précisément « l'application ». De fait, il y aura toujours une dimension logique applicative dans un client, ne serait-cte que pour savoir quoi appeler en réponse à telle ou telle commande activée par l'utilisateur·ice.

Dans le cas d'un client unique, notamment d'un client lourd comme il s'en réalisait volontier à l'époque (mais plus rarement maintenant au profit d'applications web), cela peut s'envisager. On redescend alors à 3 couches, en ayant bien en tête que le client est le siège de la logique applicative.



Mais si on ambitionne de réaliser plusieurs clients différents, alors on aura intérêt à mutualiser la logique applicative dans une couche distincte siégeant côté serveur. On aura alors bel et bien une architecture à 4 couches.