CFPP - 2021

ARCHITECTURE N-TIERS

UNIVERSITÉ GASTON BERGER DE SAINT-LOUIS

Dr Babacar Diop

Architecture multi-tiers

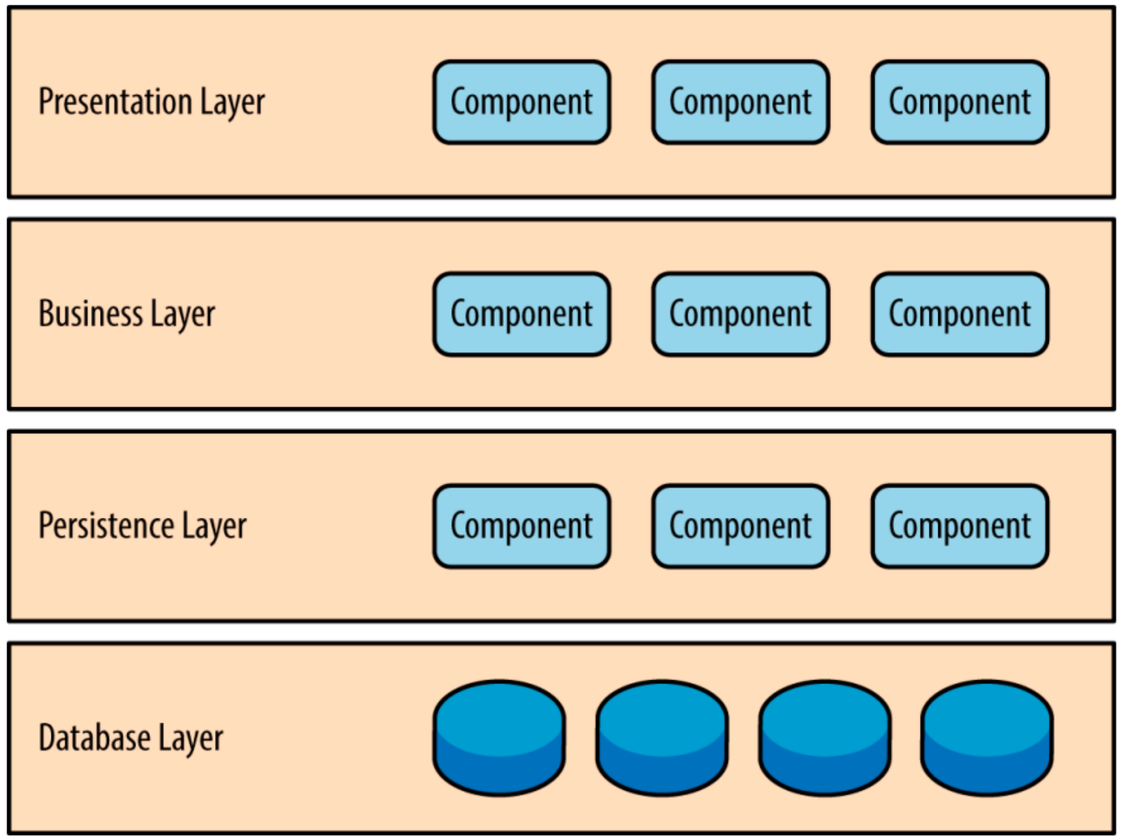
ARCHITECTURE LOGICIELLE

- L'architecture la plus répandue
- Utilisée sur la plupart des applications logicielles
- Java EE

Description

ARCHITECTURE LOGICIELLE

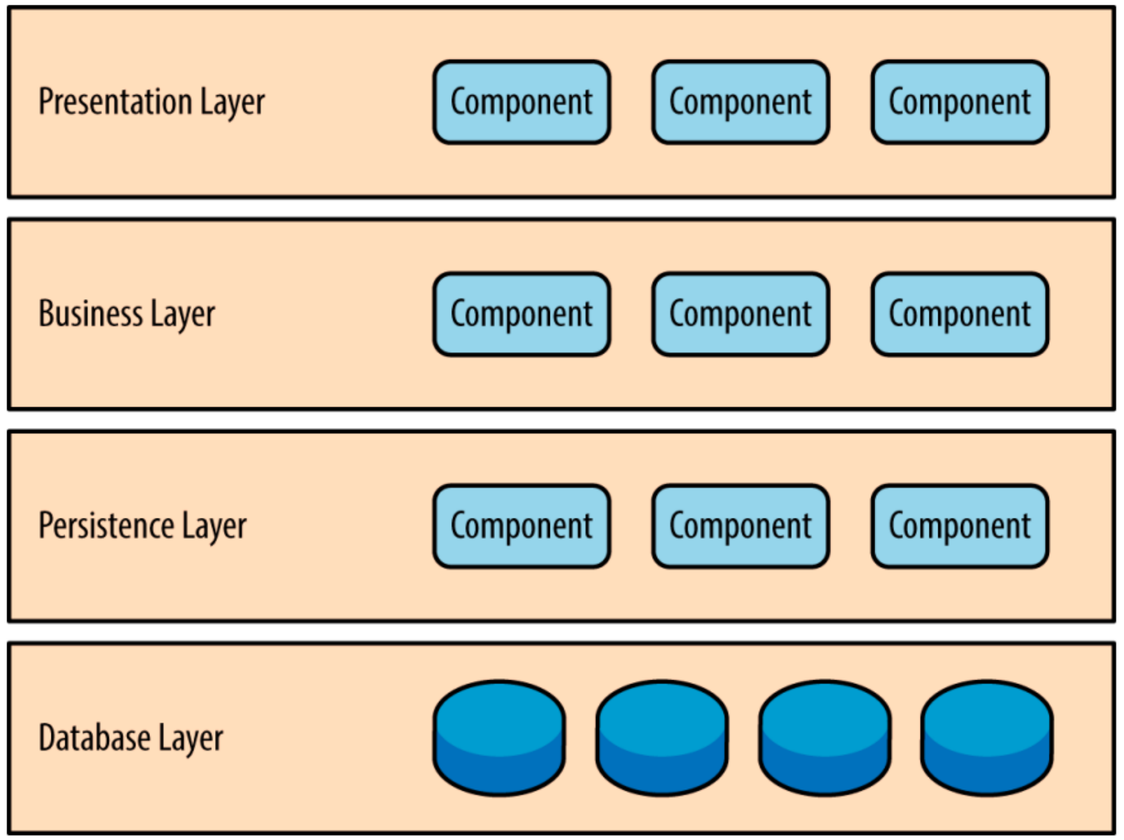
- Composants organisés en couches horizontales
- Chaque couche avec un rôle spécifique à jouer
- Ne spécifie pas le nombre de couches, mais on a 4 couches:
  - **Présentation**
  - **Business**
  - **Persistence**
  - **Base de données**



Description

ARCHITECTURE LOGICIELLE

- **Présentation** : responsable de l'affichage des données
- **Business** : se focalise sur la logique de l'application
- **Persistence** : gère l'interaction vers la base de données
- **Base de données** : fichiers d'enregistrements des données



Caractéristiques

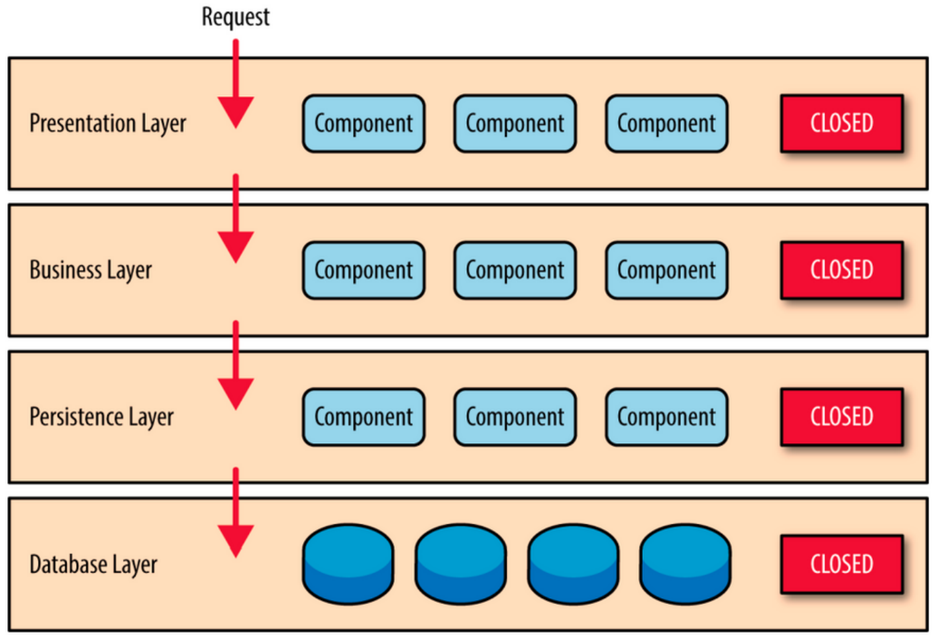
ARCHITECTURE LOGICIELLE

- Découplage entre les différents composants des différentes couches
- Les composants d'une couche spécifique gèrent la logique de cette couche
- Facilité d'attribuer des roles et des responsabilités dans l'architecture
- Facilité de développement, de tests, de gestion des différentes composantes

Concepts clés

ARCHITECTURE LOGICIELLE

- Toutes les couches sont marquées comme "**FERMÉS**"
- Ce qui veut dire que le flot de données passe entre des couches adjacentes
- Par exemple, la couche Présentation ne peut pas directement passer vers la couche Persistence sans passer par la couche Business

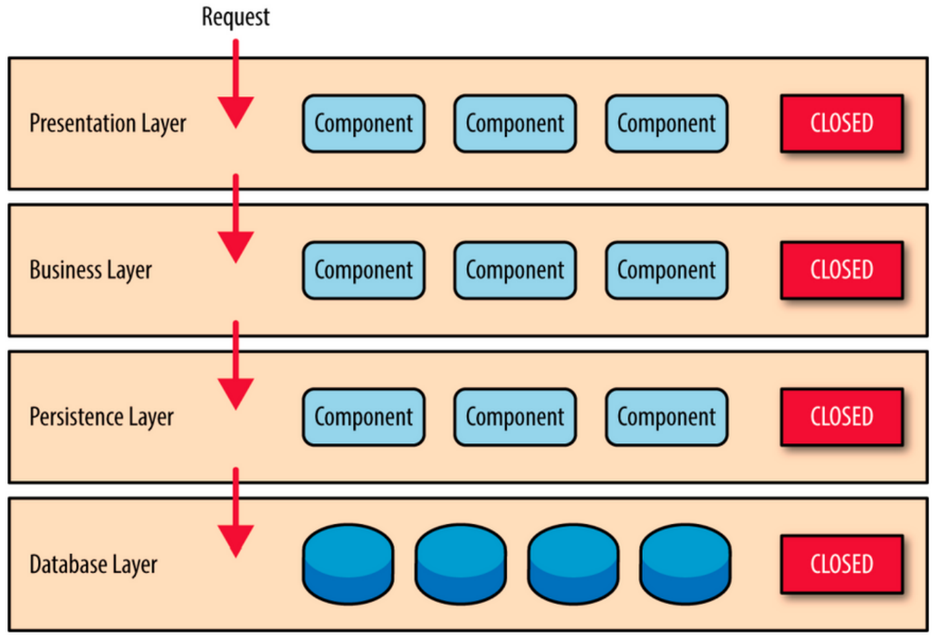


Concepts clés

ARCHITECTURE LOGICIELLE

- L'usage des couches "**FERMÉS**" facilite l'**isolation entre les différentes couches**

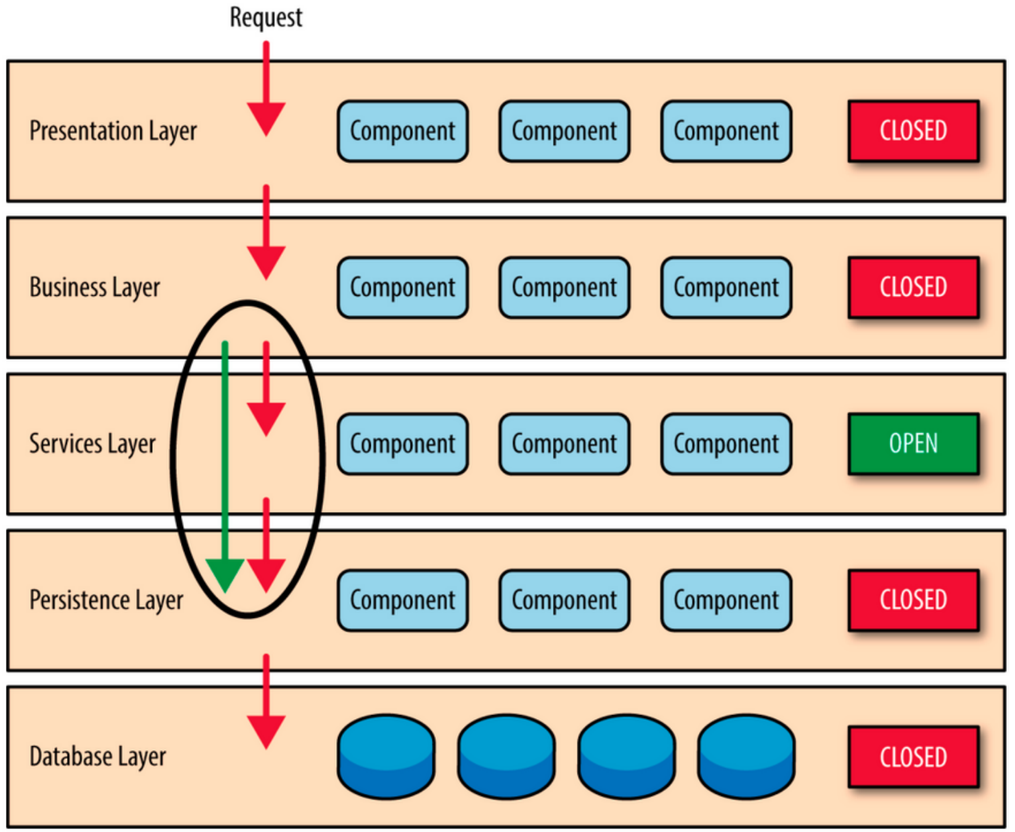
- Les couches sont indépendantes et chaque couche n'a pas de vue sur une autre couche



Concepts clés

ARCHITECTURE LOGICIELLE

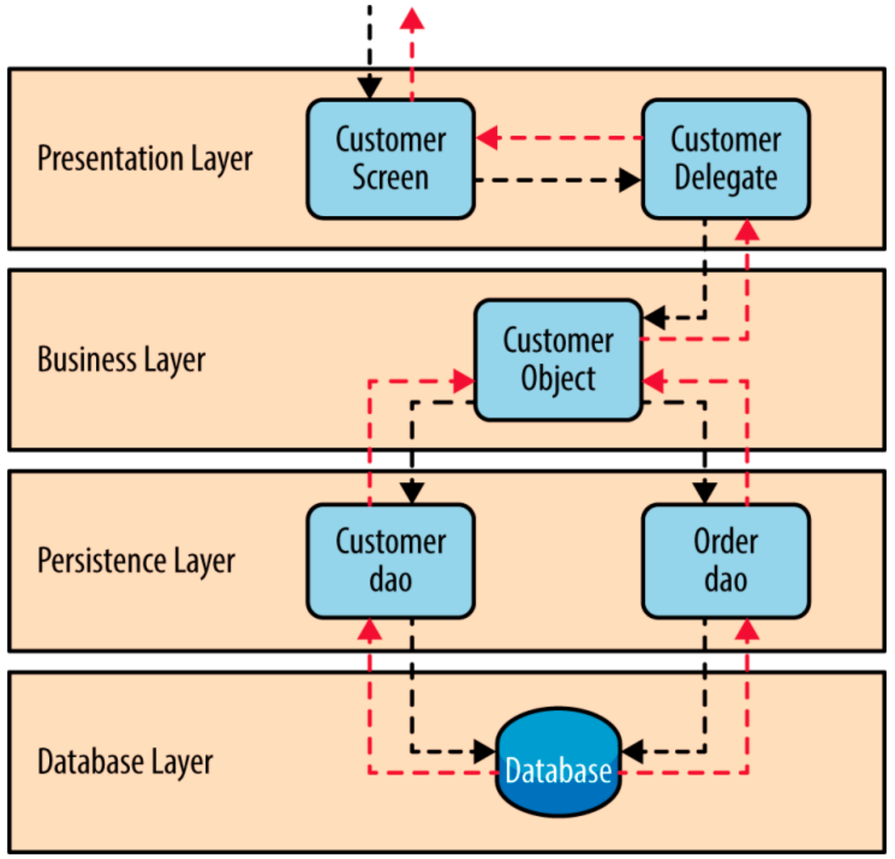
- L'usage des couches "**FERMÉS**" facilite l'**isolation entre les différentes couches**
- Les couches sont indépendantes et chaque couche n'a pas de vue sur une autre couche



Exemple de fonctionnement (1)

ARCHITECTURE LOGICIELLE

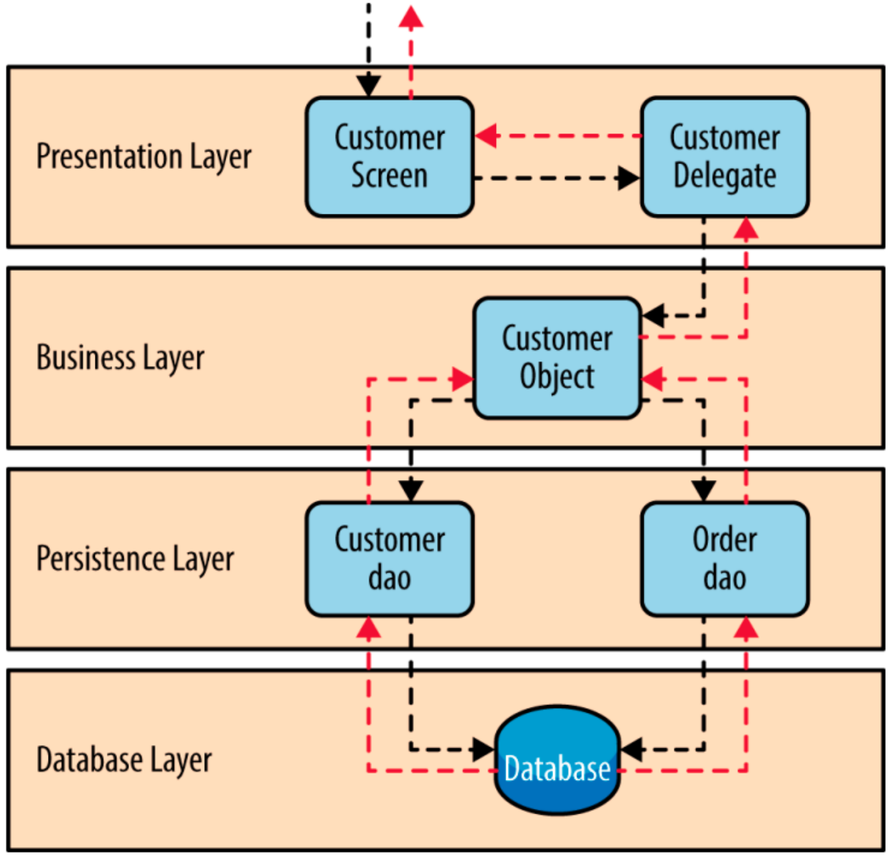
Les **flèches noires** montrent la demande descendant vers la base de données pour récupérer les données de client, et les **flèches rouges** montrent la réponse remontant vers l'écran utilisateur.



Exemple de fonctionnement (2)

ARCHITECTURE LOGICIELLE

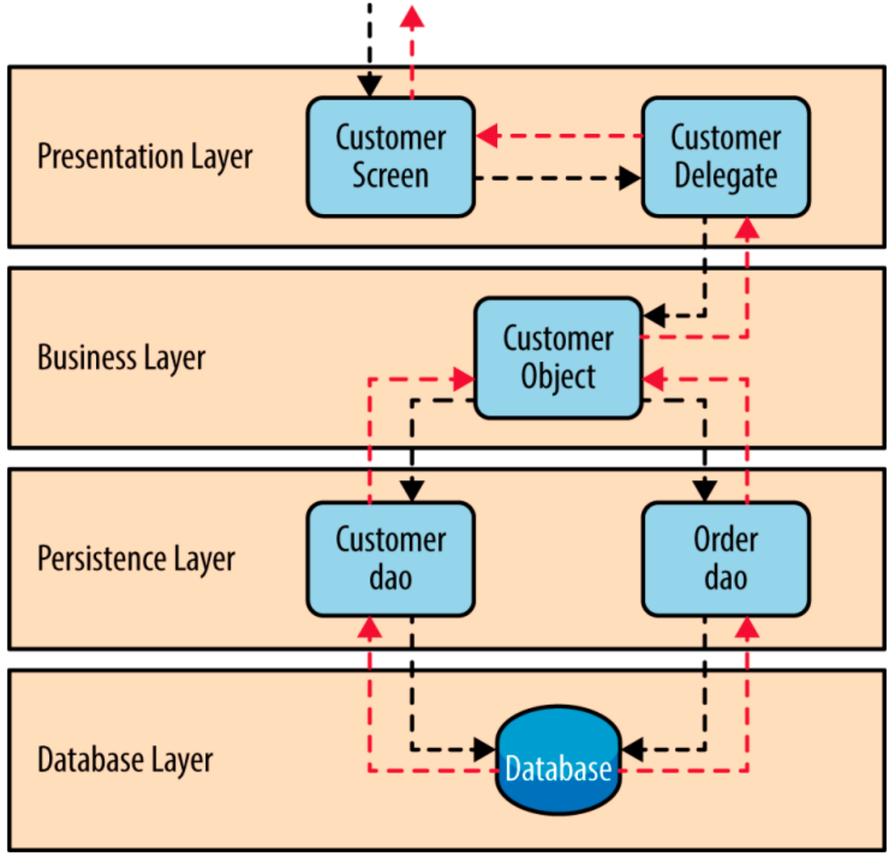
Dans cet exemple, les informations sur le client comprennent à la fois des ***données sur le client*** et des ***données sur les commandes*** (commandes passées par le client).



Exemple de fonctionnement (3)

ARCHITECTURE LOGICIELLE

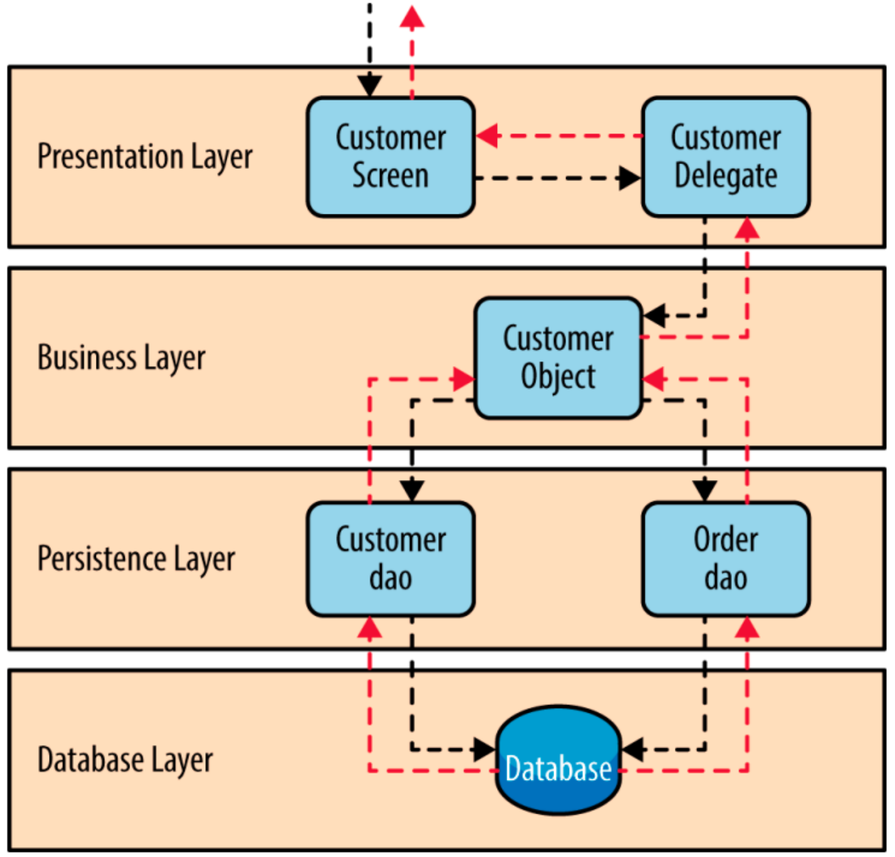
L'écran client(**Customer screen**) est responsable de l'acceptation de la demande et de la diffusion des informations sur le client.
Il ne sait pas où se trouvent les données, comment elles sont récupérées ou les tables qui doivent être interrogées pour obtenir les données.



Exemple de fonctionnement (4)

ARCHITECTURE LOGICIELLE

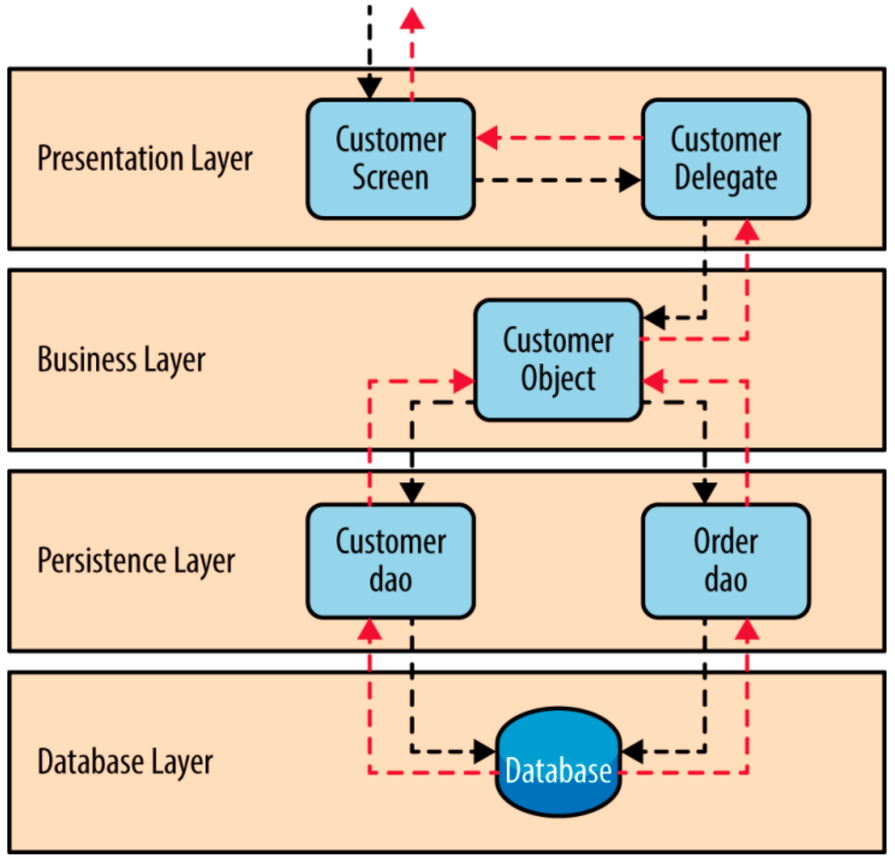
Lorsque l'**écran client** (**Customer screen**) reçoit une demande d'informations sur le client pour un individu particulier, il transmet alors cette demande au **module de délégation des clients** (**Customer delegate**).



Exemple de fonctionnement (5)

ARCHITECTURE LOGICIELLE

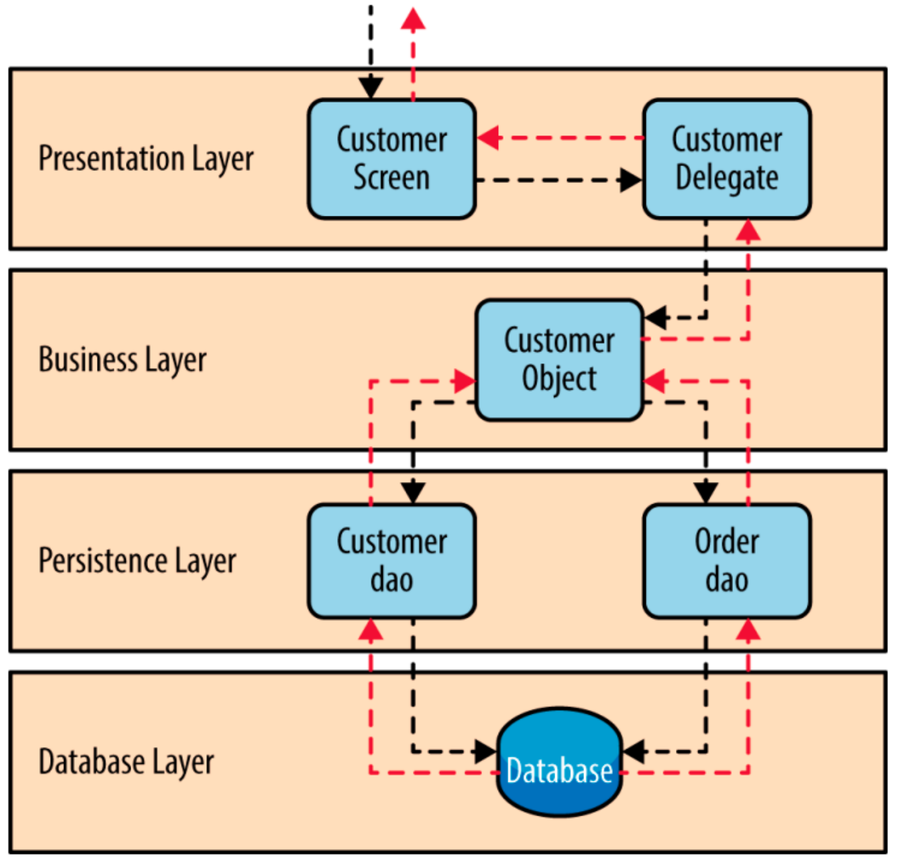
Ce module est chargé de savoir quels modules de la **couche métier** peuvent traiter cette demande, mais aussi comment accéder à ce module et quelles sont les données dont il a besoin (le contrat).



Exemple de fonctionnement (6)

ARCHITECTURE LOGICIELLE

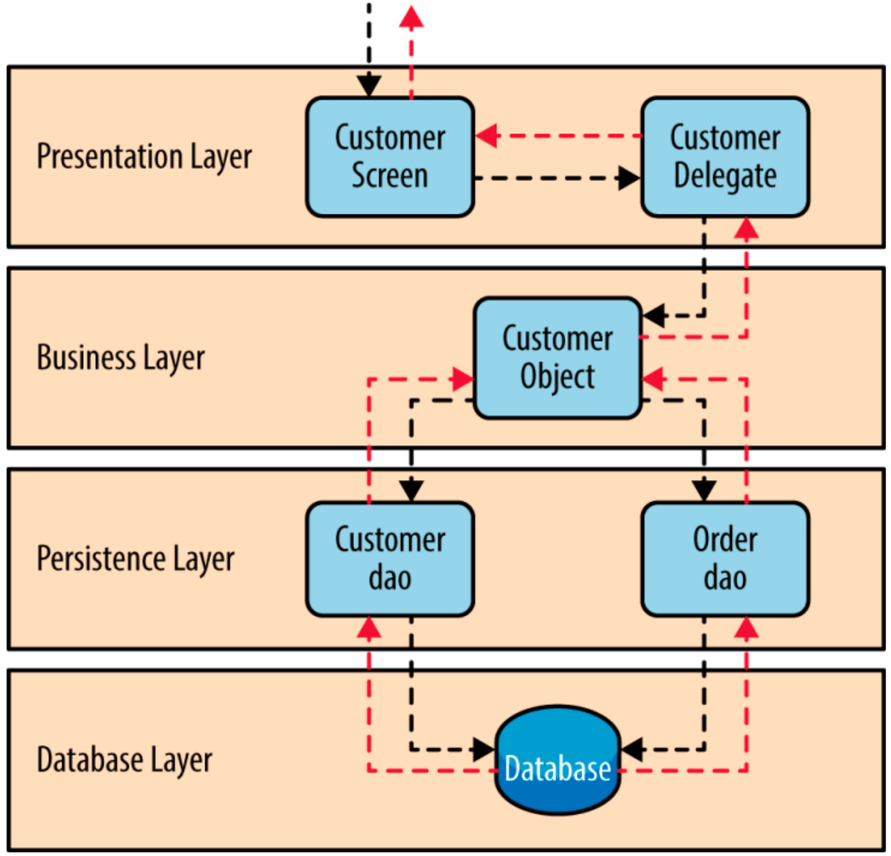
L'**objet client** (**Customer Object**)
de la couche métier
est responsable de l'agrégation de toutes les informations nécessaires à la demande de l'entreprise (dans ce cas, pour obtenir des informations sur le client).



Exemple de fonctionnement (7)

ARCHITECTURE LOGICIELLE

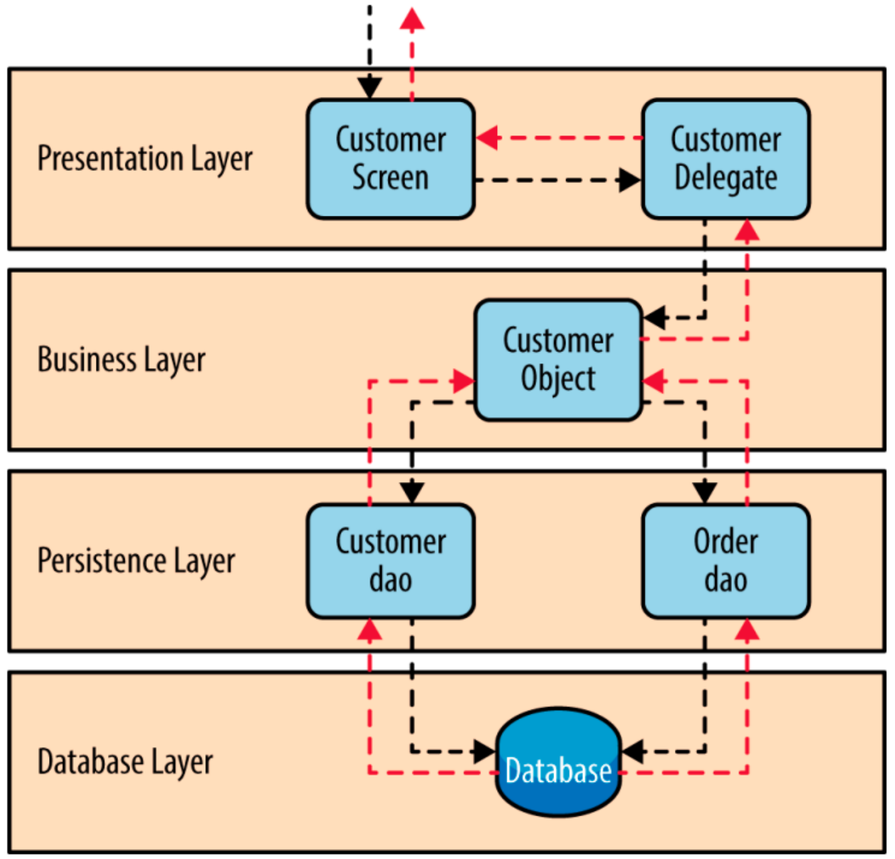
Ce module fait appel au **module Client dao** (**Customer dao**) du client dans la couche de persistance pour obtenir des données sur les clients, ainsi qu'au **module Order dao** (**Order dao**) pour obtenir des informations sur les commandes.



Exemple de fonctionnement (8)

ARCHITECTURE LOGICIELLE

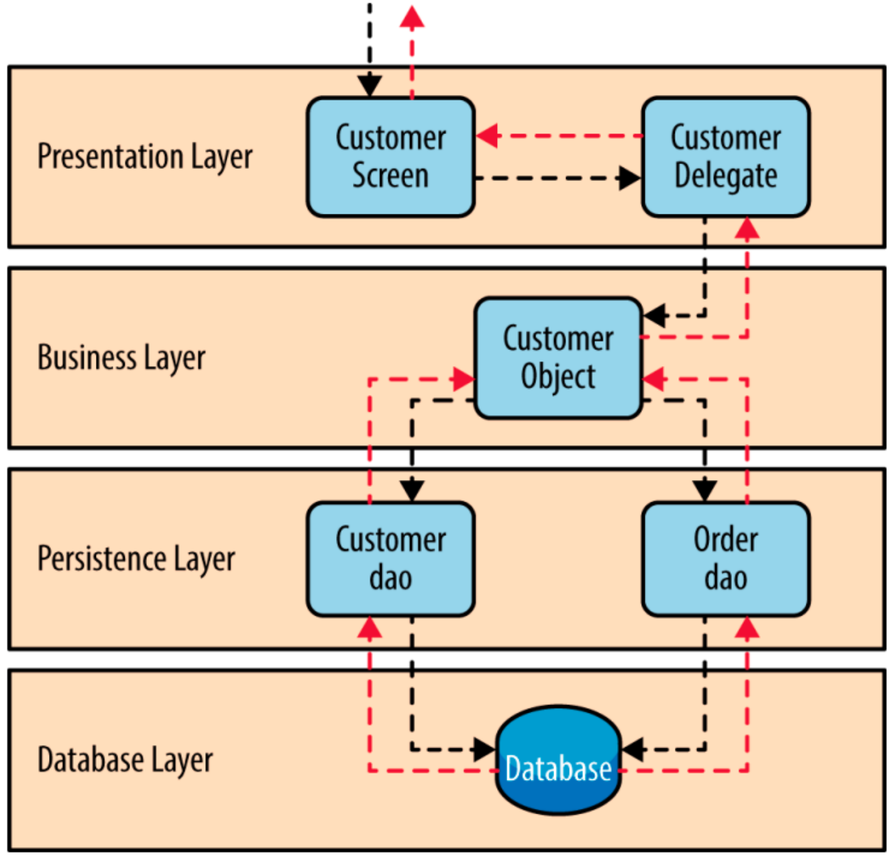
Ces modules exécutent à leur tour SQL pour récupérer les données correspondantes et les renvoyer à l'objet client dans la couche à l'objet client dans la couche métier.



Exemple de fonctionnement (9)

ARCHITECTURE LOGICIELLE

Une fois que l'objet client reçoit les données, il les agrège et transmet ces informations au délégué du client, qui transmet ensuite ces informations au délégué du client qui les transmet ensuite à l'écran client pour qu'elles soient présentées à l'utilisateur.



Implémentation

ARCHITECTURE LOGICIELLE

- **Customer screen**: Java Server Faces (JSF), ASP
- **Customer object**: Spring bean, EJB3 bean, .NET
- **Data access objects**: Plain Old Java Objects, JDBC

Considérations

ARCHITECTURE LOGICIELLE

- **Attention** aux **gouffres d'architecture !**
- **Gouffres d'architecture:** décrit une situation dans laquelle les requêtes passent plusieurs couches de l'architecture sans nécessiter des opérations spécifiques au niveau des couches de passage.

Considérations

ARCHITECTURE LOGICIELLE

- **Attention** aux **gouffres d'architecture !**
- Toute architecture aura un certain nombre de scénarios correspondant aux **gouffres d'architecture**
- L'idée c'est d'analyser le pourcentage de requêtes de cette catégorie et de le minimiser autant que possible (**20%** semble OK)

Analyse de performances

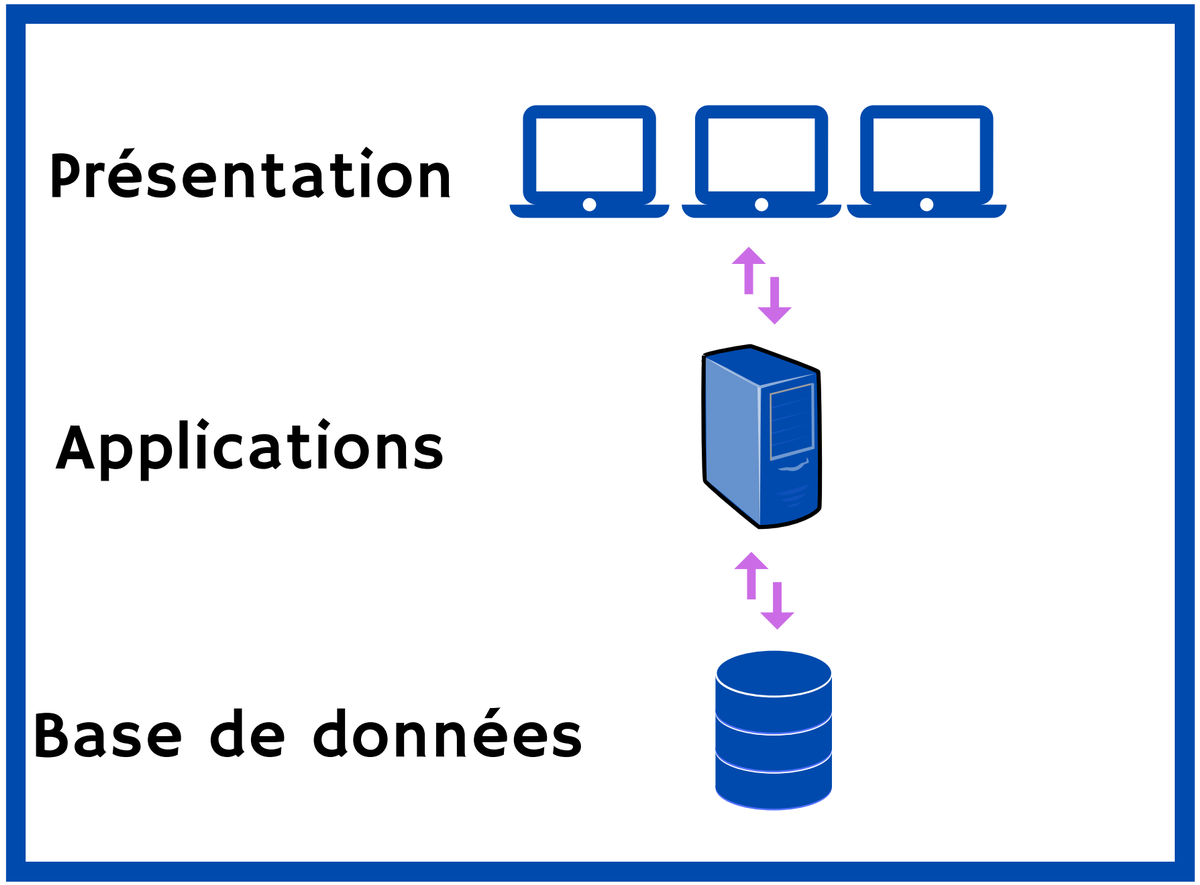
ARCHITECTURE LOGICIELLE

• Agilité	Faible
• Facilité de déploiment	Faible
• Testabilité	Fort
• Performance	Faible
• Passage à l'échelle	Faible
• Facilité de développement	Fort

Architecture 1-tier

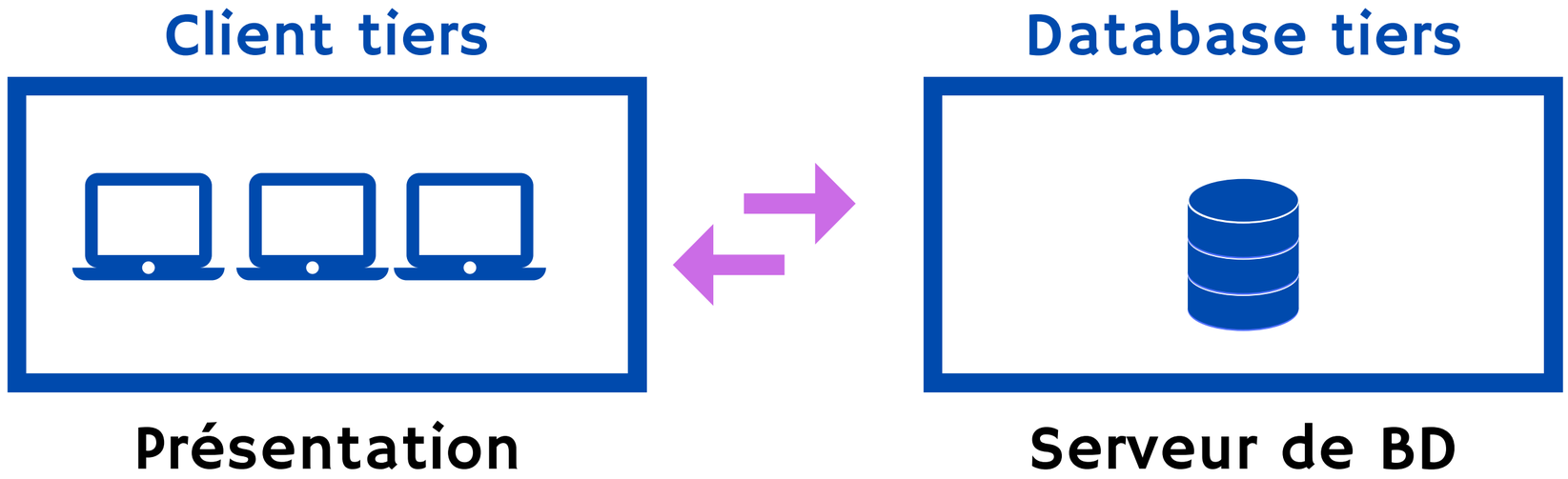
ARCHITECTURE LOGICIELLE

Client tiers



Architecture 2-tiers

ARCHITECTURE LOGICIELLE



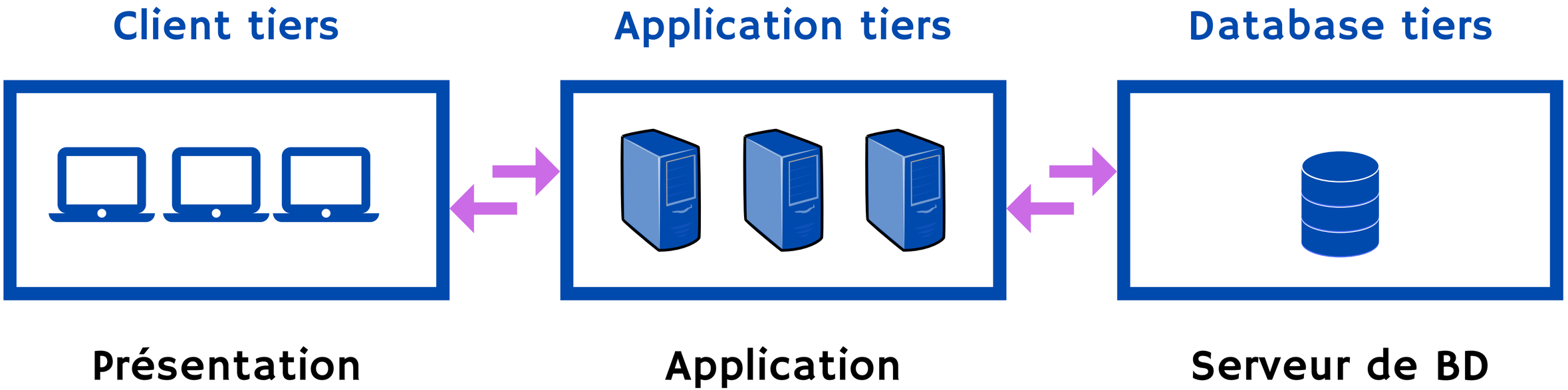
Architecture 2-tiers

ARCHITECTURE LOGICIELLE

- Client-Serveur applications
  - Procédures d'appel à distance (RPC)
  - Orienté Objet (OO) CORBA - RMI -
  - Orienté Ressource (HTTP, RESTful APIs)

Architecture 3-tiers

ARCHITECTURE LOGICIELLE



Analyse de performances

ARCHITECTURE LOGICIELLE