LOG430 – Architecture logicielle

Groupe 01

# < Performance > – Justification (en classe)

Pour une banque, un très grand nombre de clients peuvent faires des transactions sur des ATM en même temps, et un seul serveur qui reçoit ces requêtes et obtient ou met à jour le solde ou d’autres informations sur les comptes peut ne pas être suffisant pour traiter tout ce traffique.

# <Performance> – Scénario 1 (en classe)

**Source**: Le client qui utilise l’ATM

**Stimuli**: Une transaction demande de communiquer avec les serveurs de la banque

**Artéfact**: ATM et serveurs de la banque

**Environnement**: Le serveur de la banque est sollicité à un niveau qu’il supporte

**Réponse**: Le serveur répond à l’ATM que la transaction a été effectuée

**Mesure de la réponse**: Le temps nécessaire à la transaction et le nombre de transaction supporté sans dégrader le service de manière importante

# <Performance> – Tactique 1 (en classe)

**Description**: Manage ressources - Maintain multiple copies of data / – data replication

**Justification**: Avoir plusieurs copies (bases de donées) de l’information sur les comptes de chaque utilisateur, en répliquant l’Information, pour améliorer les peformances. Les bases de données peuvent être à différents emplacements géographiques pour diminuer la latence.

**Description**: Manage ressources - Maintain multiple copies of computation / – load balancer

**Justification**: Dans un même centre de données de la banque, un mécanisme de round-robin ou de distribution de diviser le traitement sur plusieurs machines, augmentant la capacité.

# <Performance> – Vue Architecturale (en classe)

**Serveur de** base de données

**Load balancer**

**Distribue (round robin)**

**Load balancer**

**Serveur de transactions**

**Serveur de transactions**

**ATM**

**Distribue (round robin)**

**Réplication**

**Envoie des transactions**

**Serveur de** base de donnés

Groupe 02

Scénarios de qualité, tactiques et justifications des attributs produits parles équipes en classe

<Performance> – Justification

La clientèle de la banque a émis plusieurs plaintes au niveau du temps de réponse des guichets. Quand il y a plusieurs personnes qui utilisent simultanément les guichets de la succursale, le temps de réponse devient vite important et peut changer le niveau d’appréciation de la banque. Une analyse par les techniciens a montré que le système de gestion de bases de données était à blâmer pour les mauvaises performances des guichets.

<Performance> – Scénario

**Source** : Client (utilisateur du guichet)

**Stimuli** : Initialisation d’une transaction

**Artéfact** : Guichet bancaire

**Environnement** : Opération normale du guichet

**Réponse** : La transaction est traitée (le client fait un dépôt, retrait, transfert ou vérifie le solde)

**Mesure de la réponse** : Mesurer le temps moyen par type de transaction

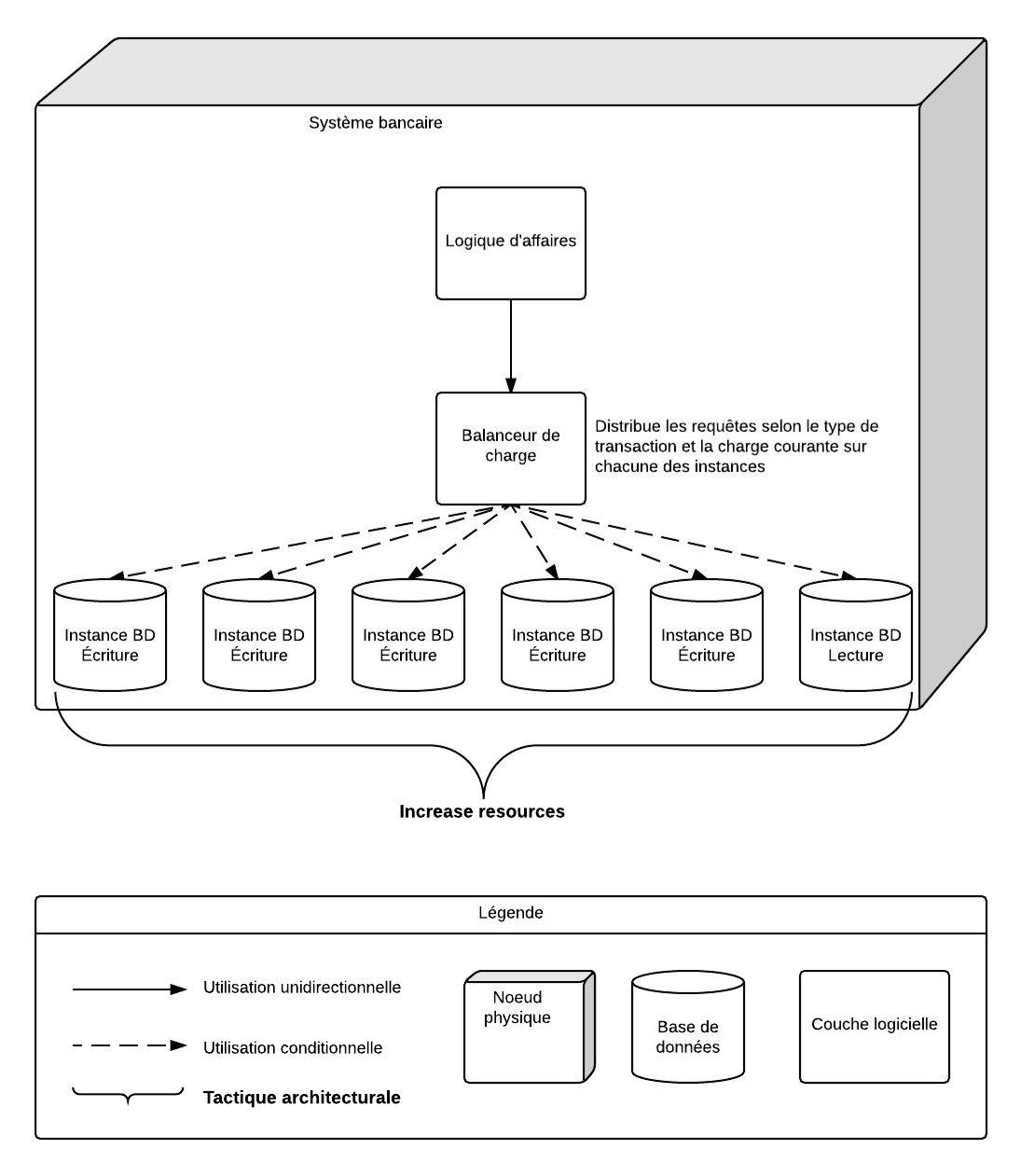
<Performance> – Tactique

**Description** : Increase resources

**Justification** : Ajouter des serveurs d’écriture en BD car le plus grand nombre de requêtes faites via le guichet sont des transaction bancaire, ce qui nécessite l’enregistrement de records et non la lecture de donnés.

<Performance> – Vue Architecturale

La vue architecturale illustre l’ajout de multiples serveurs de bases de données. La logique d’affaires communique avec un balanceur de charges qui envoie les demandes au bon serveur de bases de données et on peut facilement ajouter des ressources dans le réseau pour permettre une meilleure performance.



Éléments du diagramme

|  |  |
| --- | --- |
| **Élément** | **Description** |
| Système bancaire | Le noeud physique correspondant de manière large au système bancaire qui traite les transactions demandées dans les guichets. |
| Système bancaire / Logique d’affaires | La couche logicielle contenant toute la logique d’affaires des systèmes bancaires. |
| Système bancaire / Balanceur de charge | La couche logicielle contenant toute la logique nécessaire pour balancer les charges entre les différents serveurs de bases de données |
| Système bancaire / Instance BD Écriture | Une instance physique de bases de données qui permet l’écriture. Plusieurs sont utilisées car dans le domaine bancaire, c’est plutôt rare de devoir faire de la lecture sans écriture. |
| Système bancaire / Instance BD Lecture | Une instance physique de bases de données qui permet la lecture seulement. Il y en a moins que pour l’écriture mais ce type d’instance peut traiter les requêtes en lecture seule plus rapidement. |

La tactique *I* i*ncrease Resources* correspond essentiellement à augmenter le nombre d’éléments *Instance BD Écriture* dans le diagramme de vue architecturale présenté.

Cette tactique répond au besoin décrit dans le scénario qui consiste à augmenter la performance des bases de données. Puisque dans un système bancaire, faire du *caching* n’est pas vraiment une option, augmenter le nombre d’instances est quelque chose qui peut être coûteux mais qui permet d’avoir les mêmes fonctionnalités sans compromis.

Groupe 05

**Scénarios de qualité, tactiques et justifications des attributs produits par les équipes en classe**

**< Performance > – Justification (en classe)**

Améliorer le temps de réponse avec le serveur

**<Performance> – Scénario 1 (en classe)**

**Source**: Les algorithmes de communication avec le serveur

**Stimuli**: Vérification de la correspondance du numéro de la carte bancaire avec le NIP.

**Artéfact**: Processus de validation sur le serveur bancaire.

**Environnement**: Normal.

**Réponse**: Le serveur accepte le numéro de compte.

**Mesure de la réponse**: Temps d'attente pour la réponse du serveur.

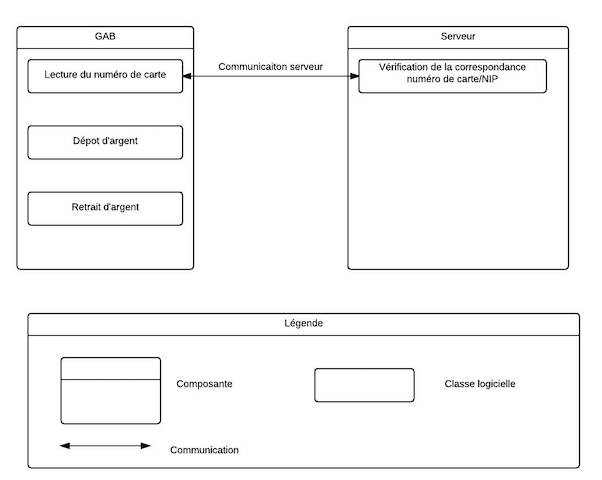
**<Performance> – Tactique 1 (en classe)**

**Description**: Réduction des temps d’exécution de l'algorithme de vérification.

**Justification**: La réponse plus rapide de l'algorithme réduit le temps d'attente de l'utilisateur final.

**<Performance> – Vue Architecturale (en classe)**

Diagramme:



Texte de description du diagramme

* Le diagramme suivant représente l’interaction qu’il y a entre un guichet automatique bancaire et un serveur. On peut remarquer que le guichet bancaire s’occupe de lire le numéro de carte qui est immédiatement envoyé au serveur pour vérification. Le serveur vérifie le NIP et rend son verdique sur la validité de celui-ci. Ensuite, l’utilisateur peut faire des dépôts, retraits,etc.

Table de description des éléments du diagramme

* **GAB**: Guichet automatique bancaire que le client utilise pour effectuer un retrait.
* **Serveur**: Serveur de la banque qui gère toute l’information des clients.
* **Lecture du numéro de carte**: Classe logicielle qui gère toute la lecture de la carte.
* **Dépôt d’argent**: Classe logicielle qui s’occupe de réaliser des dépôts au guichet automatique.
* **Retrait d’argent**: Classe logicielle qui s’occupe de réaliser des retraits au guichet automatique.
* **Vérification de la correspondance numéro de carte/NIP**: Classe logicielle du serveur qui vérifie la validité du NIP/numéro de carte pour ainsi effectuer des opérations.

Texte décrivant la relation entre les éléments et la tactique

* La classe «Lecture du numéro de carte» fait la lecture du numéro de carte pour ensuite envoyer les informations au serveur, à la classe «Vérification de la correspondance numéro de carte/NIP». En améliorant cette dernière, son algorithme, nous pourrons réduire le temps d’attente de l’utilisateur et accélérer le processus de transaction.

Groupe 06

**< Performance > – Justification (en classe)**

Les clients sont exigeants en ce qui a trait aux performances des guichets automatiques, lors d’un retrait, un client ne veut pas voir un time out. Une trop grande attente peut stresser le client et ainsi dégrader son expérience utilisateur.

**<Performance> – Scénario 1 (en classe)**

**Source**: Les utilisateurs du guichet automatique

**Stimuli**: Les utilisateurs opèrent une transaction (retrait) dans le guichet automatique

**Artéfact**: Serveur client (serveur qui vérifie que le client a bien le montant qu’il veut retirer)

**Environnement**: Operational mode

**Réponse**: L’approbation de la transaction donnée par le serveur

**Mesure de la réponse**: Stimuli ­ réponse < 2 secondes

**<Performance> – Tactique 1 (en classe)**

**Description**: Manage Resources : Maintain Multiple copies of Data et Increase Resources

**Justification**: En ayant plusieurs copies des données des comptes client sur plus de serveurs, notre processus transactionnel aura une réponse beaucoup plus facile puisque les copies des données sont accessibles en lecture, et la réponse à savoir si le retrait est possible peut être renvoyée plus rapidement, et par la suite, après l’envoi de la réponse, on peut aller mettre la BD à jour.

**<Performance> – Vue Architecturale (en classe)**

* ∙  Diagramme
* ∙  Légende
* ∙  Texte de description du diagramme
* ∙  Table de description des éléments du diagramme
* ∙  Texte décrivant la relation entre les éléments et la tactique
* ∙  texte décrivant comment la tactique répond a chacun des scénarios de qualité.

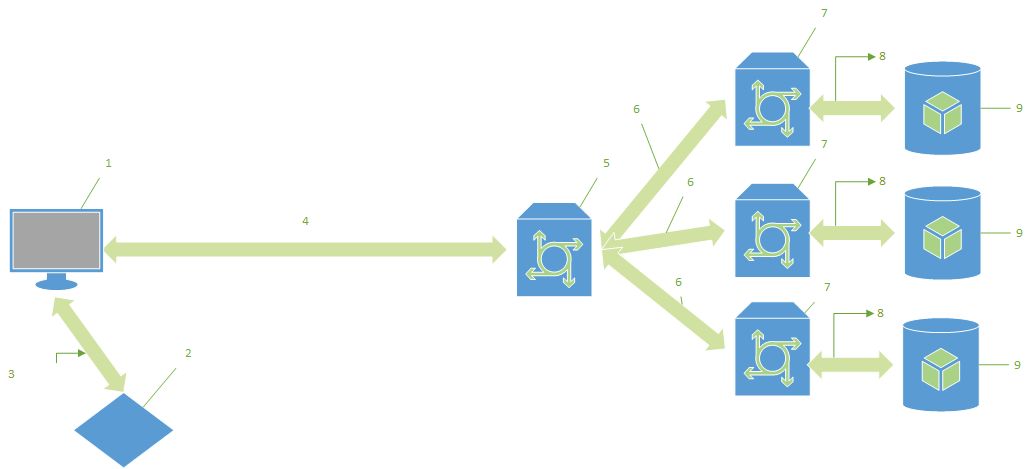


Diagramme illustrant les modules acteurs lors d’un retrait bancaire au guichet automatique. La communication avec le serveur de gestion de la charge et la redirection aux divers serveurs identiques selon la charge de travail.

**Relation entre les éléments et la tactique**

Dans ce diagramme, il est possible de noter les canaux utilisés afin de gérer la charge sur les divers serveurs d’accès aux données bancaires. Les serveurs et bases de données bancaires possèdent les mêmes données et peuvent donc fournir la même réponse à la même requête.

**Réponse au scénario de qualité**

La réponse peut donc être plus rapide en divisant le travail côté serveur. Les temps de réponse ciblés peuvent donc être maintenus lors d’un achalandage important.

|  |  |
| --- | --- |
| **Élément** | **Description** |
| 1 | Interface du guichet automatique |
| 2 | Module d’analyse de la quantité d’argent liquide disponible |
| 3 | Communication pour la validation du solde d’argent liquide disponible |
| 4 | Protocole https : communication guichet­serveur de gestion de charge |
| 5 | Serveur de gestion de charge |
| 6 | Communication avec les divers serveurs bancaires |
| 7 | Serveur bancaire |
| 8 | Communication avec la base de données bancaire |
| 9 | Base de données bancaire |

Groupe 07

**< Performance > – Justification (en classe)**

La performance du système GAB est importante, car il faut que celui­ci réponde dans des temps acceptables. Les clients vont cesser d’utiliser le système s’il n’est pas assez rapide.

**<Performance> – Scénario ­ Temps de réponse d’une transaction (en classe)**

**Source**: La source est externe; il s’agit du temps de réponse du serveur lorsqu’on effectue une transaction

**Stimuli**: Le stimuli provient d’événements sporadiques; les transactions ne sont pas régulières

**Artéfact**: Le module de communication du GAB

**Environnement**: Dans un environnement normal, le délai d’attente doit être court

**Réponse**: On reçoit le résultat de la transaction en réponse

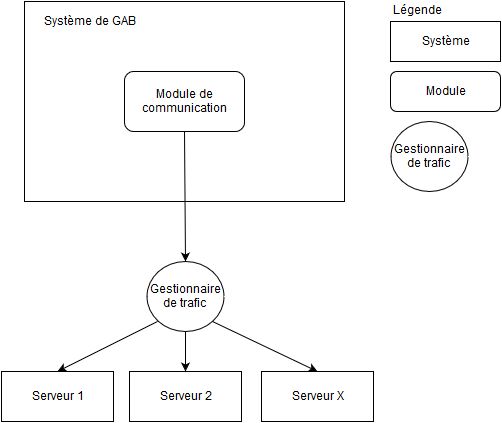
**Mesure de la réponse**: La latence entre l’envoi de la transaction et la réception de la réponse

**<Performance> – Tactique 1 (en classe)**

**Description**: “Maintain multiple copies of computations”

**Justification**: Le guichet envoie sa requête dans un système de gestion de trafic qui renvoie la requête au serveur le moins occupé. Ainsi, chaque serveur a une charge de travail semblable et ne risque pas d’être surchargé et le temps de réponse est constant pour l’ensemble des demandes. On peut assigner un système de gestion de trafic à un certain nombre maximal de GAB pour éviter qu’il devienne le point de ralentissement.

Vue Architecturale



Le module de communication envoie une transaction vers le gestionnaire de trafic. Celui­ci connaît la charge des serveurs auxquels il est assigné, il peut donc envoyer la transaction vers le serveur le moins occupé. Cela permet d’équilibrer la charge des transactions sur chacun des serveurs disponibles. Puisque le gestionnaire de trafic devient un goulot lui­même, il faut établir sa charge maximale et ne pas lui assigner plus de GAB qu’il ne peut en traiter.

La tactique est d’utiliser plusieurs ressources pouvant effectuer le traitement de la requête. Nous avons aussi ajouté un système permettant de séparer les charges entre chacune de ces ressources pour que le temps de réponse soit uniforme.

Groupe 10

**<Performance> – Justification (en classe)**

Le GAB doit être performant. Le temps de réponse doit répondre selon les normes établies dans le document B651.1-F09 (C2015) - Conception accessible des guichets automatiques bancaires.

**<Performance> – Scénario 1 (en classe)**

**Source**: Système

**Stimuli**: Le temps de réponse de l’interface pour faire un retrait ne répond pas à la norme définit dans le document B651.1-F09.

**Artéfact**: Code

**Environnement**: Environnement de maintenance

**Réponse**: Amélioration du temps de réponse pour effectuer un retrait.

**Mesure de la réponse**: Temps de réponse en millisecondes.

**<Performance> – Tactique 1 (en classe)**

**Description**: Increase Resource

**Justification**: L’ajout de processeur et une connexion internet plus rapide aurait une impact direct sur le temps de réponse de l’interface. De plus, c’est la façon la plus rapide d’avoir un résultat.

**<Performance> – Vue Architecturale (en classe)**