Scénarios, tactiques et vues architecturales

Performances

[**1 À partir du moment ou un retrait s'effectue sur le GAB jusqu'au moment où l'argent est sorti du GAB, moins de 5 secondes s'est écoulé**](#_ir0q76nql4nc) **3**

[1.1 Tactique 1 : Prioritize events](#_cn2zqyz35ck0) 3

[1.2 Tactique 2 : Maintain multiple copies of computations](#_30j0zll) 3

[1.3 Vue architecturale (à l’extérieur de la classe)](#_jgrtner1zv82) 4

[1.3.1 Description du diagramme](#_1fob9te) 4

[1.3.2 Description des éléments](#_3znysh7) 4

[1.3.3 Relation entre éléments et tactiques](#_2et92p0) 4

[**2 Scénario : Améliorer la performance quand l’usager fait une transaction bancaire**](#_nu4it8fd8u59) **5**

[2.1 Tactique 1 : Maintenir une copie locale des informations client.](#_g4lw3kmtjgvg) 5

[2.2 Tactique 2 : Améliorer les ressources matérielles](#_o3p3116vik6z) 5

[2.3 Vue architecturale (à l’extérieur de la classe)](#_1wpfsohyj8tu) 6

[**3 On veut qu’un client puisse effectuer un transfert entre 2 de ses comptes.**](#_sjl8c1nofvic) **7**

[3.1 Tactique 1 : load balancing instance la moins occupée prend en charge la requête](#_gks4abe2vai0) 7

[3.2 Tactique 2 : multiple copies of data](#_o5k5cn7oyd89) 7

[3.3 Tactique 3 : Schedule resources](#_wi8c666svcwu) 7

[3.4 Diagramme tactique 1:](#_ecpy97mu6ae7) 8

[3.4.1 Description du diagramme](#_mb1u5tcjbdq) 9

[3.4.2 Table de description des éléments du diagramme](#_73h92kqiod4j) 9

[3.4.3 Description de la relation entre les éléments et les tactiques](#_z1jq6tgyftxy) 10

[**4 Scénario : Le premier du mois, le service de dépôt est beaucoup plus achalandé. Provoquant des ralentissements(Surtout à St-Jérôme, Lachute, Joliette).**](#_d5rpkh7drpcz) **11**

[4.1 Tactique 1 : Introduce Concurrency](#_8oqg0l1poqm) 11

[4.2 Tactique 2 : Increase Ressource](#_8byz4xx4nmg2) 11

[4.3 Tactique 3 : Maintain multiple copies of computation](#_low7i1hri51) 11

[4.4 Vue architecturale (à l’extérieur de la classe)](#_jtpdys9ffdsb) 12

[4.4.1 Diagramme](#_tyjcwt) 12

[4.4.2 Texte de description du diagramme](#_8jjmwfy02gyl) 12

[4.4.3 Texte de description des éléments du diagramme](#_3dy6vkm) 12

[4.4.4 Texte décrivant la relation entre les éléments et les tactiques](#_co0bs8pb4v49) 13

[**5 Scénario : Performance**](#_uwum9g31h7je) **14**

[5.1 Tactique 1 :](#_552qjd9igboz) 14

[5.2 Vue architecturale (à l’extérieur de la classe)](#_73uvi5f2welt) 15

[**6 Scénario : Performance**](#_t9llbenwaosf) **16**

[6.1 Tactique 1 : Control Resources Demand - Increase resource efficiency](#_xip3c3etyxzv) 16

[6.2 Tactique 2 : Manage Resources - Increase resources](#_498nv8fqwh5x) 16

[6.3 Tactique 3 : Manage Resources - Introduce concurrency](#_gmx8ofv9by7u) 17

[6.4 Vue architecturale (à l’extérieur de la classe)](#_u0rg6oyohy1t) 18

[6.4.1 Diagramme](#_4d34og8) 18

[**7 Scénario : Latence d’authentification client**](#_iq4nt9ar0my5) **19**

[7.1 Tactique 1 : Augmenter l’efficacité des ressources](#_wbc3cr4fwaay) 19

[7.2 Tactique 2 : Prioriser l’événement](#_hthgmg2lnho6) 19

[**8 Scénario :**](#_8qroras6vffy) **20**

[8.1 Tactique 1 : Priorisation des événements](#_thjg12h6atzc) 20

[8.2 Tactique 2 : Concurrence](#_rngdxuqs0nbn) 20

[8.3 Vue architecturale (à l’extérieur de la classe)](#_sf4fokudo4uw) 21

[8.3.1 Diagramme](#_2s8eyo1) 21

[8.3.2 Légende](#_infzbvhljkx9) 21

[8.3.3 Texte de description du diagramme](#_352170qgrokm) 21

[8.3.4 Table de description des éléments du diagramme](#_cw6woak63szb) 21

[8.3.5 Texte décrivant la relation entre les éléments et les tactiques](#_ms9dntuo72o7) 22

# 1 À partir du moment ou un retrait s'effectue sur le GAB jusqu'au moment où l'argent est sorti du GAB, moins de 5 secondes s'est écoulé

|  |  |
| --- | --- |
| **objectifs d'affaires** | Un retrait devrait s’effectuer en moins de 5 secondes. |
| **Source** | Externe (Client) |
| **Stimulus** | Événement de retrait stochastique |
| **Artéfact** | Composante de transactions du système |
| **Environnement** | Mode d’opération normal |
| **Réponse** | Le GAB remet l’argent au client. |
| **Mesure de la réponse** | Latence à effectuer le retrait. |
| **Questions** | 1. Quel est le temps de retour des divers systèmes bancaires face aux retraits? |

## **1.1 Tactique 1 : Prioritize events**

**Description**: Un système de gestion central recevant toutes les requêtes des différents GAB traitera les retraits avant les dépôts.

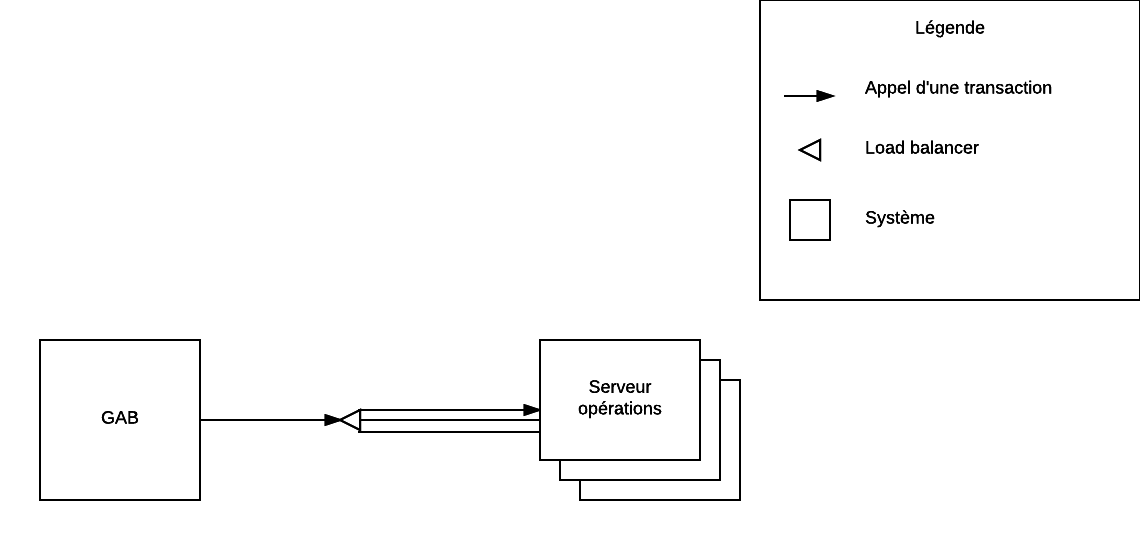
**Justification**: Puisque le système de retrait doit s'effectuer plus rapidement que le dépôt, il est donc nécessaire de prioriser l'événement.

## **1.2 Tactique 2 : Maintain multiple copies of computations**

**Description**: Suite à la priorisation des transactions, un load balancer se chargera de transmettre les opérations au serveur concurrent moins chargé.

**Justification**: Avec un load balancer, le système de transaction ne devra pas attendre que le serveur se libère avant d'effectuer une autre transaction et sera redirigé vers le premier serveur libre.

## **1.3 Vue architecturale (à l’extérieur de la classe)**



### **1.3.1 Description du diagramme**

Cette vue composants et connecteurs démontre les interactions entre le GAB et les serveurs d’opérations pour améliorer la performance.

### **1.3.2 Description des éléments**

**GAB**: Représente le système du Gab.

**LoadBalancer**: Représente le load balancer du système.

**Serveur opérations**: Représente les serveurs d'opérations

### **1.3.3 Relation entre éléments et tactiques**

**Tactique Prioritize events:**

Le load balancer s'occupera de prioriser les opérations de retrait envoyé au serveur d'opération.

**Tactique Maintain multiple copies of computations:**

Le load balancer s’occupera de gérer les multiples serveurs d’opérations, qui permettent d’effectuer les transactions en simultanées de manière plus rapide.

# **2 Scénario : Améliorer la performance quand l’usager fait une transaction bancaire**

|  |  |
| --- | --- |
| **Objectifs d’affaires** | Réduire le temps de réponse d’un retrait de 25 % afin d’offrir une meilleure expérience à l’utilisateur. |
| **Source** | Développeur |
| **Stimulus** | * ~~Mauvais résultat lors d’un sondage auprès des clients.~~ * Temps moyen de transaction trop élevé. |
| **Artéfact** | * ~~Code~~ * Réseau |
| **Environnement** | * ~~Lors d’une~~ opération bancaire normale |
| **Réponse** | * ~~Tester la performance de réponse actuelle.~~ * ~~Modifier la performance de réponse.~~ * ~~Tester la performance de réponse~~ * ~~Déployer.~~ |
| **Mesure de la réponse** | * Effectuer un autre sondage de la satisfaction de la clientèle. * Tests de performance interne. |
| **Questions** | * Est-ce qu’en améliorer la performance ça augmente les coûts d’opération ? * Est-ce qu’il y a un écart-type défini ? |

## **2.1 Tactique 1 : Maintenir une copie locale des informations client.**

**Description**: Conserver plusieurs copies de l’information, ce qui permet de traiter l’information localement.

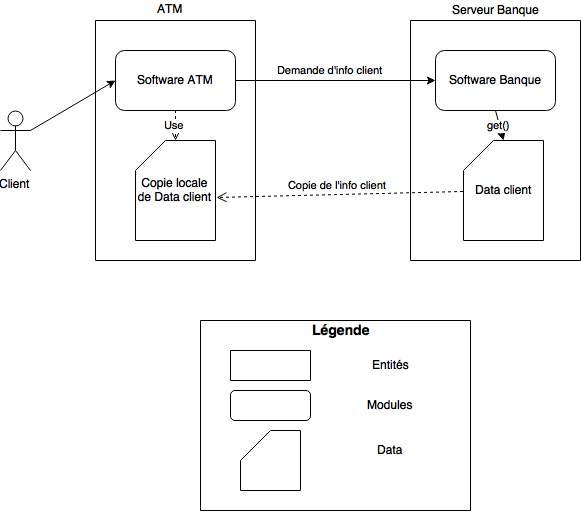
**Justification**: Au début de la séance, toutes les informations sont sauvegardées localement. Une fois que toutes les transactions sont effectuées, le système bancaire va communiquer avec le serveur en envoyant toutes les données sauvegardées en cache.

## **2.2 Tactique 2 : Améliorer les ressources matérielles**

**Description**: Augmenter la performance des machines.

**Justification**: L’augmentation des ressources va permettre d’améliorer la vitesse de traitement de l’information locale. En jumelant cette tactique avec la première, nous n’avons pas à communiquer avec le serveur et le traitement local sera rapide.

## **2.3 Vue architecturale (à l’extérieur de la classe)**



**Composantes :**

**ATM:**

Le Shell ATM, le système bancaire va conserver plusieurs copies locales de client. Une fois que les transactions sont terminées, le système bancaire va communiquer avec le serveur en envoyant toutes les données sauvegardées en local.

**Serveur Banque :**

Au début de la transaction, le serveur va fournir les informations du client qui désire de faire une transaction bancaire. Ainsi, il doit attendre que le système bancaire aille traiter toutes les transactions avant d’y recevoir.

# 3 On veut qu’un client puisse effectuer un transfert entre 2 de ses comptes.

|  |  |
| --- | --- |
| **objectifs d'affaires** | La banque veut que le client puisse faire des opérations sur ses différents comptes dans un délai acceptable environ 2 secondes. |
| **Source** | Le client qui veut effectuer un transfert |
| **Stimulus** | un transfert entre deux comptes de clients de la banque |
| **Artéfact** | Le système ATM , Connexion réseau de la banque et banque de données de la banque |
| **Environnement** | Operationnal mode : normal |
| **Réponse** | Les transactions ont étés effectuées |
| **Mesure de la réponse** | Le temps de latence d’un transfert ne doit pas dépasser 2 secondes  ~~Le changement devrait prendre~~ |
| **Questions** | 1. Combien de temps doit prendre un transfert entre deux comptes avant d’être terminé? 2. Entre quels genres de comptes peut-on faire des transferts d’argent? 3. Peut-on transférer de l'argent dans un autre de nos comptes dans une autre banque? 4. Quelle est la limite du montant que l’on peut transférer? |

## **3.1 Tactique 1 : load balancing instance la moins occupée prend en charge la requête**

Description: Un load balancer répartira les requêtes équitablement de manière à ne pas surcharger un serveur. Le load balancer est capable de détecter si un serveur répond moins vite ou est momentanément indisponible afin d’envoyer les requêtes sur un autre serveur.

**Justification**: Cela devrait permettre des requêtes d’au maximum 2 secondes……...

## 3.2 Tactique 2 : multiple copies of data

**Description**: Plusieurs serveurs en réplication master-master, dont certains seront utilisés seulement pour la lecture et d’autres pour l’écriture des données.

**Justification**:Si un serveur est indisponible, un autre pourra prendre sa place automatiquement sans perte de données, la première requête pourrait être de plus de 2 secondes, le temps qu’un autre serveur prenne la place de celui qui est surchargé ou indisponible. Plusieurs serveurs en lecture, car la banque doit afficher beaucoup d’information sur l’ATM et a seulement besoin d’écrire en base de données pour les dépôts.

## 3.3 Tactique 3 : Schedule resources

**Description**: Lors des périodes de pointe(jeudi soir le jour de paie), le système alloue plus de ressources aux ATM afin qu’il puisse faire les transferts d’argent entre les différents comptes du client plus rapidement.

**Justification**: Si le système a plus de ressources lors des périodes de pointe, il pourra répondre rapidement et efficacement aux nombres grandissants d’utilisateurs . Cela permet ainsi de réduire le stress sur le système sans affecter la qualité du service. Après la période de pointe, l’allocation de ressources redevient normale.

## 3.4 Diagramme tactique 1:



**Diagramme tactique 2:**



 : Serveur SQL master

**Légende**

: Lien de communication

### 3.4.1 Description du diagramme

Le diagramme est une représentation exhaustive du réseau entre l’ATM et les serveurs.

Dans le diagramme de la tactique #1, l’ATM communique directement avec un load balancer qui aura la charge de monitorer les serveurs SQL (charge et temps de réponse), afin de rediriger les requêtes vers le serveur qui possède la plus petite charge ou temps de réponse, ce qui va permettre d’améliorer le temps de réponse d’une requête.

Dans le diagramme de la tactique #2, l’ATM communique directement sur un point d’accès qui enverra les requêtes sur un des serveurs du cluster et recevra une réponse pour la renvoyer à l’ATM. Chaque serveur SQL du cluster est en master-master ce qui permet d’avoir une synchronisation identique entre tous les serveurs SQL et permettra au point d’accès d’utiliser un serveur SQL qui est moins “demandé” afin d’améliorer le temps de réponse. C’est un mélange entre load balancing (le point d’accès) et la réplication des données (le cluster).

### 3.4.2 Table de description des éléments du diagramme

|  |  |
| --- | --- |
| **Élément** | **Description** |
| ATM | L’ATM est la représentation physique de notre système bancaire. Le client utilise ce terminal afin de faire des opérations bancaires comme un dépôt, un retrait ou un transfert d’argent entre 2 compte. |
| Load balancer | C’est ce qui permet de distribuer également les nombreuses requêtes envoyées par notre ATM vers nos différents clusters de serveurs. |
| Cluster de serveur | Les serveurs qui vont traiter les différentes requêtes envoyées par l’ATM et redistribuées par le load balancer |
| Sql access point | Le point d’accès pour accéder à nos différentes bases de données. |

### 3.4.3 Description de la relation entre les éléments et les tactiques

Selon notre diagramme, le load balancer permet de redistribuer toutes les différentes requêtes efficacement entre nos différents cluster de serveurs. De plus, à l'aide de la tactique de ressources scheduling, le système permet d’allouer des ressources supplémentaires durant les périodes de pointe. Le système peut ainsi servir les clients plus rapidement et efficacement tout en respectant un temps de latence raisonnable. Il s’agit de deux secondes dans notre cas. De plus, grâce à la technique de multiple copies of data, la performance de notre système sera améliorée , car le système pourra écrire les données dans une base de données et les lire dans une autre.

# 4 Scénario : Le premier du mois, le service de dépôt est beaucoup plus achalandé. Provoquant des ralentissements(Surtout à St-Jérôme, Lachute, Joliette). La transaction devrait se faire en moins de 5 secondes

|  |  |
| --- | --- |
| **objectifs d'affaires** | Répondre à la demande augmentée le premier du mois. |
| **Source** | Externe |
| **Stimulus** | Une plus grosse demande de dépôt le premier du mois. |
| **Artéfact** | Les serveurs centraux de la banque. |
| **Environnement** | Peak load |
| **Réponse** | On génère plus de machine Docker afin de répondre à la demande.  Passage de Peak load à Over load.  Traiter la demande de retrait |
| **Mesure de la réponse** | Une transaction doit prendre en moin de 5 secondes.  Le nombre de transactions du NbGuichets/Minutes  Le nombre de transaction en attente  La charge de chaque serveur. |
| **Questions** | 1. Quel est la vitesse du processeur ? 2. Combien de coeur possède le processeur ? 3. ~~Quel est la vitesse de l’internet en upload et en download ?~~ 4. ~~Quel est la capacité de stockage d’enveloppe de dépôts ?~~ 5. Combien y a-il de guichets disponible au public ? |

## 4.1 Tactique 1 : Introduce Concurrency

**Description**: Ajout de machines docker afin de répondre à la demande.

**Justification**: Afin d’ajouter de la performance, on peut ajouter des machines pour augmenter la capacité de traitement du système.

## 4.2 Tactique 2 : Increase Ressource

**Description**: Améliorer les performances matériels du serveur. Ajout de mémoire vive, de puissance des processeurs.

**Justification**: Puisque le serveur ne parvient pas à contenir la demande lors d’une forte achalandage du service, il faut améliorer les spécifications de celui-ci.

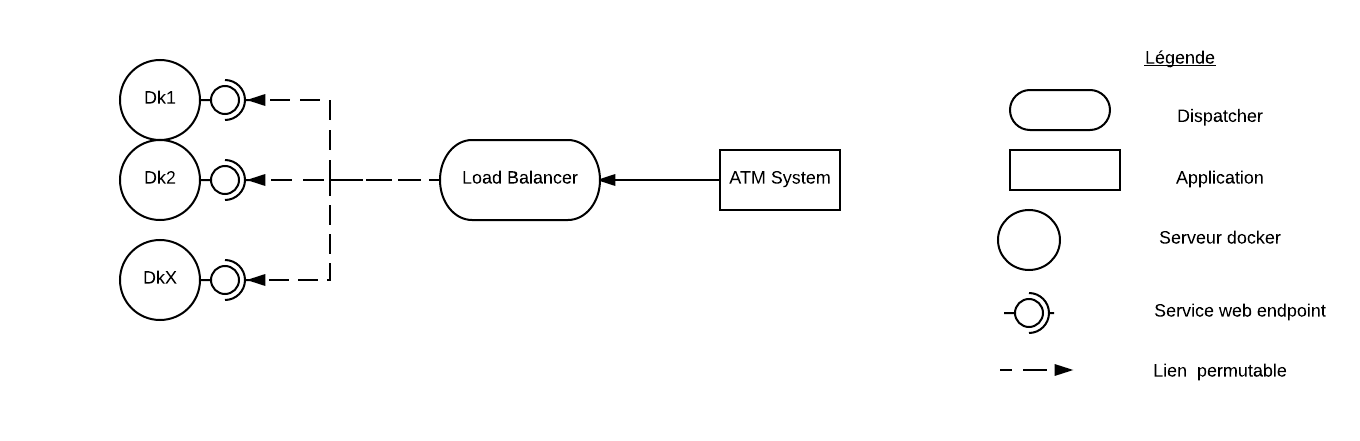
## 4.3 Tactique 3 : Maintain multiple copies of computation

**Description**: Le fait d’utiliser plusieurs machines Docker permet de cloner les machines afin d’augmenter la performance.

**Justification**: En ayant plusieurs files et en utilisant un load balancer, on peut répartir les tâches.

## 4.4 Vue architecturale (à l’extérieur de la classe)

### 4.4.1 Diagramme



### 4.4.2 Texte de description du diagramme

Cette vue représente globalement comment les requêtes du services de dépôts sont traité pendant une période très occupé. En effet, atm system va envoyer la requête au load balancer qui est une sorte de dispatcher. Il aura pour rôle de transférer la requête à un serveur docker qui n’est pas trop chargé, Si les serveurs de base sont occupé, alors il va envoyer un signal vers les machines docker pour cloner de nouveau serveurs et ainsi répondre à la demande.

### 4.4.3 Texte de description des éléments du diagramme

|  |  |
| --- | --- |
| Élément | Description |
| ATM system | Le système principal de la caisse qui gère les dépôts, ,retraits, transfert... |
| Load balancer | Système qui va s'occuper de répartir les tâches entre les différents serveurs. |
| Dk | Machine docker qui vont traiter la demande |
| lignes pointillé | liens dynamique qui peut être créé vers une nouvelle machine docker si tous les autres serveurs actifs sont surchargé. |

### 4.4.4 Texte décrivant la relation entre les éléments et les tactiques

La tactique introduce currency représente l’ajout des machines docker qui sont nommé DkX dans le diagramme. Le increase ressource se fait aussi au niveau des machines docker ou un technicien aura acheté et installé des meilleures composants pour avoir une meilleure performance. Enfin, le maintain multiple copies of computation est représenté par le fait qu’on a ajouté un loadbalancer qui va traiter plusieurs requêtes en même temps et va les acheminer dans le serveur le moins occupé.

# 5 Scénario : Performance

|  |  |
| --- | --- |
| **objectifs d'affaires** | Permettre d’offrir des services rapides et sans délai. |
| **Source** | Utilisateur |
| **Stimulus** | Retrait d’argent |
| **Artéfact** | Système |
| **Environnement** | Exécution normale |
| **Réponse** | Le système réponds et distribue l’argent demandée |
| **Mesure de la réponse** | Le système mets moins de deux secondes à communiquer avec le serveur et l’argent est disponible dans le distributeur en moins de 5 secondes, pour un total de 7 secondes suite à la demande initiale. |
| **Questions** | 1. Y a-t’il un montant maximum de billets que l’on peut retirer du guichet? 2. Ce peut-il que dans certains secteurs il y ait des délais réseau pouvant affecter le temps de réponse? 3. Ce peut-il que pendant certains moments de la journée la charge sur le serveur est plus élevée et affecte le temps de réponse négativement? 4. Quel est le temps de réponse maximal qui serait acceptable pour un client moyen? 5. Existe-t’il des technologies de communication alternatives pouvant améliorer le temps de réponse du système qui sont à portée budgétaire? |

## 5.1 Tactique 1 :

**Description**: Introduire de la concurrence système

**Justification**: Pour réduire le temps total d’attente d’un client pour une transaction donnée, certaines informations communiquées avec le serveur peuvent être envoyées d’avance pour permettre une réponse rapide. Par exemple, lorsqu’un utilisateur désire effectuer un retrait, le serveur est avisé et il effectue l'autorisation d’avance pour le compte, jusqu’à un certain montant. Par la suite, lorsque l’utilisateur confirme la transaction, celle-ci est pré-approuvée et le serveur n’attends que la confirmation de la part du GAB pour retirer le montant indiqué.

## 5.2 Vue architecturale (à l’extérieur de la classe)

### 

La vue ci-dessus représente une vue “Pipe & Filter”, de la famille composants et connecteurs. Cette vue permet de représenter les traitements issus de l’application de la tactique d’introduction de concurrence système. En effet, nous pouvons voir les processus qui sont effectués sur le GAB(la ligne du dessus) ainsi que les processus qui sont effectués sur le serveur de la banque(la ligne du dessous).

# 6 Scénario : Performance

|  |  |
| --- | --- |
| **Objectifs d'affaires** | Le système doit donner l’argent entre 5 et 8 secondes par tranche de 100$ lors d’un retrait. |
| **Source** | Guichet Automatique Bancaire. |
| **Stimulus** | Demande de retrait d’argent rapide. |
| **Artéfact** | Le guichet, le système de gestion de compte. |
| **Environnement** | Normal. |
| **Réponse** | Connexion entre le compte utilisateur et la banque.  Confirmer la disponibilité dans le compte du client.  Décompte de l’argent. |
| **Mesure de la réponse** | Le temps de communication avec le réseau de la banque.  Le temps de traitement de débitation dans le compte client.  Le temps de décompte de billets à donner.  Le montant moyen d’argent retiré. |
| **Questions** | 1. Quelle est la vitesse de décompte mécanique des billets? 2. Avez-vous un gros trafic de transaction? 3. Quel protocole de communication est utilisé entre les guichets et les banques? 4. Quel est le temps d’attente moyen accepté par les clients? 5. Quel est le montant moyen des retraits effectués par les client? |

## 6.1 Tactique 1 : Control Resources Demand - Increase resource efficiency

**Description**: Il est possible d'améliorer l’algorithme et de réduire le nombre de fonctions pour augmenter la rapidité de réponse du système.

**Justification**: Par exemple, pour notre système, si on réduit le nombre de vérification et d’appels aux serveurs, on peut augmenter significativement la rapidité du système. Car ce qui est le ‘bottleneck’ du système est la communication entre le client et le serveur; par la suite, c’est seulement un calcul mécanique et la transmission au client. Alors, si on peut réduire le temps de traitement des ressources, on augmente la vitesse de réponse.

## 6.2 Tactique 2 : Manage Resources - Increase resources

**Description:** Il est possible d’augmenter la performance en améliorant les processeurs, la mémoire et les communications par réseaux.

**Justification:** Pour le système, il serait possible d’améliorer la technologie de la communication réseau pour avoir une réponse beaucoup plus rapide du serveur et ainsi continuer le traitement de la demande. Ensuite, il pourrait être bon d’améliorer les composantes mécaniques, comme les compteurs de billets et faire le traitement plus rapidement aussi. Aussi, il est possible d’avoir une meilleure technologie dans les communications entre les différents composants du guichet pour qu’ils se répondent plus rapidement et avec plus d’efficacité.

## 6.3 Tactique 3 : Manage Resources - Introduce concurrency

**Description:** Permettre le traitement en parallèle de différentes actions. Permettre de distribuer les actions à faire en même temps sur différents fils d’exécution.

**Justification:** Dans notre guichet, il serait possible de faire un traitement en parallèle entre le décompte de l’argent et la demande au serveur. Il pourrait de vérifier l’argent disponible et faire le décompte en avance pendant qu’il communique avec le serveur si le client peut faire ce retrait. Donc, une fois que le guichet reçoit l’autorisation, celui-ci donne directement l’argent au client. Sinon, il remet tout simplement l’argent dans le coffre et demande au client de recommencer. On peut alors sauver beaucoup de temps entre la demande et la réception de l’argent du client.

## 6.4 Vue architecturale (à l’extérieur de la classe)

### 6.4.1 Diagramme

Performance.png

**Figure 1.0 :** Diagramme du scénario de performance.

- Le diagramme illustré dans la figure 1.0 consiste à fournir une stratégie pour améliorer la performance au niveau du comptage de billets. La stratégie consiste à établir une concurrence entre la communication du système au serveur et le décompte. Cette concurrence asynchrone va permettre d’utiliser les ressources maximum de la mémoire et du processeur, c’est pour cela que nous suggérons dans notre stratégie l’amélioration des ressources physiques afin d’améliorer le temps de réponse. de calcul et de traitement, sans oublier de faire un minimum d’appels au serveur qui causent inévitablement un délai (malgré la performance des ressources) car cela dépend évidemment de la vitesse de la connexion Internet.

**Table de description des éléments du diagramme**

|  |  |
| --- | --- |
| **Élément** | **Responsabilité** |
| Système (GAB) | Le GAB entretient des relations entre le serveur et le compteur de billet |
| Serveur | Ressource externe qui informe le GAB des informations bancaires du client |
| Compteur billet | Machine physique qui compte le nombre de liquidité. |
| Super Ordinateur | Ordinateur conçu pour atteindre les plus hautes performances. |
| Décompte | Fonctionnalité implémentée pour décompter de l’argent. Ce logiciel est embarqué sur le compteur physique. |

**Table de description des relations des tactiques entre ces éléments**

|  |  |
| --- | --- |
| **Élément** | **Description** |
| Concurrency | Traitement en parallèle du décompte et de la communication au serveur. |
| Increase Resource | Augmentation des capacités des ressources physiques, va de paire avec la concurrence. Permet de paralléliser plus efficacement. |
| Increase Resource efficiency | Non applicable dans le diagramme |

**Description des relations entre les éléments**

Le guichet utilise un traitement en parallèle entre le serveur et son compteur physique en utilisant la tactique Concurrency, ceci va permettre d’utiliser les capacités des ressources physiques à leur maximum. La tactique Increase Resources consiste tout simplement à améliorer la performance physique du système. Finalement, le système utilise la tactique Increase Resource efficiency pour réduire le nombre d’appels au serveur pour but d’augmenter significativement la rapidité de la communication client-serveur.

# 

# 7 Scénario : Latence d’authentification client

|  |  |
| --- | --- |
| **objectifs d'affaires** | Améliorer le temps de réponse entre l’ATM et la banque lors de la validation d’une carte de débit |
| **Source** | L’utilisateur de l’ATM |
| **Stimulus** | ~~L’utilisateur~~ entre sa carte et saisi son NIP |
| **Artéfact** | Le système |
| **Environnement** | Mode d’opération normal |
| **Réponse** | L’ATM initie une communication avec la banque pour vérifier le compte client. |
| **Mesure de la réponse** | Le temps nécessaire pour confirmer à l’utilisateur qu’il est bel et bien authentifié, ou qu’il a saisi un mauvais NIP. |
| **Questions** | 1. Combien de systèmes sont mis à l’oeuvre pour la validation d’une carte? 2. Quel est la priorité de l’authentification de la carte ? |

## 7.1 Tactique 1 : Augmenter l’efficacité des ressources

**Description**: Minimiser le nombre de systèmes mis à l’oeuvre pour la validation d’une carte.

**Justification**: En minimisant le nombre de systèmes mis à l’oeuvre pour la validation d’une carte, on évite des temps de traitement non nécessaires. Par exemple, si on détecte le plus tôt possible que la carte de l’utilisateur provient d’une autre banque, on peut transférer la requête à la banque concernée et éviter de tenter d’authentifier la carte contre notre base de données d’utilisateur.

## 7.2 Tactique 2 : Prioriser l’événement

**Description**: Prioriser l’authentification de l’utilisateur avant tout autre type de vérification avec le serveur de la banque.

**Justification**: En priorisant l’authentification, la latence pour s’authentifier sera réduite permettant ainsi à l’utilisateur de recevoir un réponse plus rapide serveur. Alors qu’il est possible d’afficher à l’utilisateur qu’un transfert d’argent est en traitement, et le laisser faire d’autres opérations lors du traitement, il n’est pas possible de lui afficher que l’authentification est en traitement, et de le laisser faire d’autres opérations. Ceci justifie donc la priorisation des authentifications.

# 8 Scénario :

|  |  |
| --- | --- |
| **objectifs d'affaires** | Un usager doit pouvoir être authentifié dans un délai maximal de 3 seconds après avoir entré son NIP. |
| **Source** | Source externe au système, requête au serveur |
| **Stimulus** | Il s’agit d’un événement aléatoire car on ne sait jamais quand le client arrivera |
| **Artéfact** | La connexion réseau avec les serveurs de la banque |
| **Environnement** | Mode d’opération normal |
| **Réponse** | L’événement d’authentification d’un usager doit être accepté ou refusé |
| **Mesure de la réponse** | ~~Du point de vue du guichet,~~ la latence avec le serveur d’authentification ~~doit être faible, et la demande doit être traitée en 5 secondes ou moins.~~  Le débit, ~~quant-à lui est très faible; un seul utilisateur à la fois (pour le guichet).~~ |
| **Questions** | 1. Combien d’essais pour le NIP du client? 2. Que faire si le serveur ne répond pas? |

## 8.1 Tactique 1 : Priorisation des événements

**Description**:

**Justification**: Les demandes d’authentification doivent être priorisées au niveau du guichet comparativement aux autres requêtes qui pourraient survenir au même moment, pour la maintenance, par exemple.

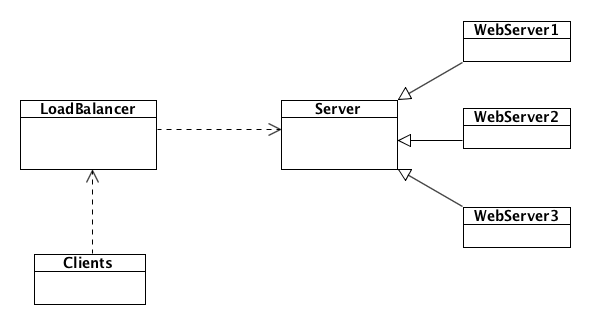
## 8.2 Tactique 2 : Concurrence

**Description**:

**Justification**: Augmenter le nombre de serveurs pouvant répondre à des demandes d’authentification de la part des guichets

## 8.3 Vue architecturale (à l’extérieur de la classe)

### 8.3.1 Diagramme



### 8.3.2 Légende

* La classe *Clients* fait appel à la classe *LoadBalancer*
* *LoadBalancer* utilise la classe *Server* qui est la mère de *WebServer1*, *WebServer2* et *WebServer3*

### 8.3.3 Texte de description du diagramme

L’implémentation d’une classe *LoadBalancer* sera incontournable. Cette dernière servira à répartir les tâches (processus de tentatives de connexion du client) entre les différents serveurs. La classe *Server* implémentera les fonctions communes des différents *WebServer.*

### 8.3.4 Table de description des éléments du diagramme

### 

|  |  |
| --- | --- |
| **Élément** | **Description** |
| Server | Classe regroupant de façon générale l’ensemble des méthodes communes aux serveurs de la banque. |
| LoadBalancer | Classe de répartition de charges, servant essentiellement à distribuer le travail entre les différentes machines (serveurs). |
| WebServer1, WebServer2, WebServer3.. | Différents serveurs de la banque |
| Clients | Interface d’authentification utilisée par l’usager de l’ATM |

### 8.3.5 Texte décrivant la relation entre les éléments et les tactiques

**Tactique 1 : Priorisation des évènements**

La classe Server aura comme rôle de prioriser les demandes d’authentification au niveau du guichet. Elle fera par le fait même, la distinction entre les requêtes d’authentification par exemple et celles portant sur la maintenance.

**Tactique 2 : Concurrence**

Le LoadBalancer, consiste à distribuer une tâche à un pool de serveurs afin premièrement de lisser le trafic réseau, mais surtout de s’assurer de la disponibilité des données recherchées, en n'envoyant des données qu'aux équipements (serveurs) en mesure de répondre, voire à ceux offrant le meilleur temps de réponse.

# **Scénario : Latence de réponse après un retrait d'argent**

|  |  |
| --- | --- |
| **objectifs d'affaires** | Optimiser le temps de réponse entre l’ATM et la banque lors du retrait d'argent. |
| **Source** | Usager de l'ATM |
| **Stimulus** | ~~L'usager fait un~~ retrait d'argent dans l'ATM |
| **Artéfact** | Le système |
| **Environnement** | Mode d'opération normal |
| **Réponse** | L'ATM donne l'argent à l'usager |
| **Mesure de la réponse** | Le temps requis pour donner l'argent à l’utilisateur. |
| **Questions** | 1. Quelle sorte de billet va être donné à l'utilisateur?  2. Quel est le montant maximum?  3. Combien de retrait peut effectuer l'utilisateur? |

## 

## **Tactique 1 : Prioriser l’événement**

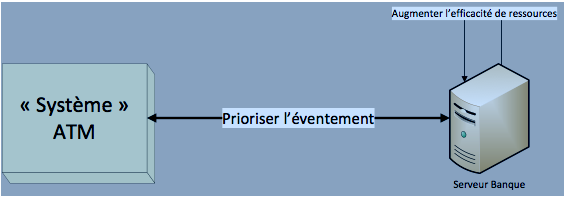
**Description**: Prioriser le traitement de retrait d'argent de l’utilisateur.

**Justification**: permets de diminuer la latence pour effectuer un retrait d'argent, permettant ainsi à l’utilisateur de recevoir une réponse plus rapide serveur.

## **Tactique 2 : Augmenter l'efficacité des ressources**

**Description**: Réduire le nombre de systèmes qui interagissent pour traiter un retrait d'argent.

**Justification**: En réduisant la quantité de systèmes qui interagissent pour le retrait d'argent des utilisateurs, on empêche des temps perdus de traitement.

****

### Diagramme 1.0 Modélisation de la performance du système d'ATM.

### Description du diagramme:

Deux composants du système, l’ATM et le serveur de la Banque ont des relations de performance en cas de problèmes. Les deux systèmes respecteront la tactique de priorisation d'éventements, ainsi que le serveur de la banque implémentera la tactique d'augmentation de l'efficacité de ressources au moment du retrait d'argent dans le système.

### Table de description des éléments du diagramme :

|  |  |
| --- | --- |
| **Tactiques** | **Descriptions** |
| Prioriser l'éventement | Prioriser le retrait d'argent |
| Augmenter l'efficacité de ressources | Permettre d'assurer le fonctionnement du retrait d'argent |

### Relations :

La ligne sur le serveur de la banque représente la tactique d'augmentation de l'efficacité de ressources..

La ligne de l'ATM vers le serveur représente la tactique de priorisation d'éventements.