LOG430 – Architecture logicielle

Équipe 02

# <Utilisabilité> – Justification

Une personne daltonienne veut utiliser le guichet bancaire automatique pour faire des transactions mais l’interface usager est aux couleurs de la banque (rouge et bleu). Ces couleurs n’étant pas différenciables par ce client, celui-ci a de fortes difficultés à lire et donc à effectuer son retrait.

# <Utilisabilité> – Scénario

**Source :** Utilisateur daltonien

**Stimuli :** Adapter le système pour personnes daltoniennes

**Environnement :** Exécution normale

**Artéfact :** L’interface utilisateur du guichet bancaire

**Réponse :** Changement des couleurs de l’interface, permettant une utilisation simplifiée pour les daltoniens

**Mesure de la réponse :**

* Nombre de minutes pour réaliser chaque type de transaction sur le guichet
* Nombre d’erreurs commises par transaction

# <Utilisabilité> – Tactique 1

**Description :** *Maintain user model* – Garder, dans le profil de l’utilisateur relié à sa carte de guichet, l’information qui indique qu’il est daltonien.

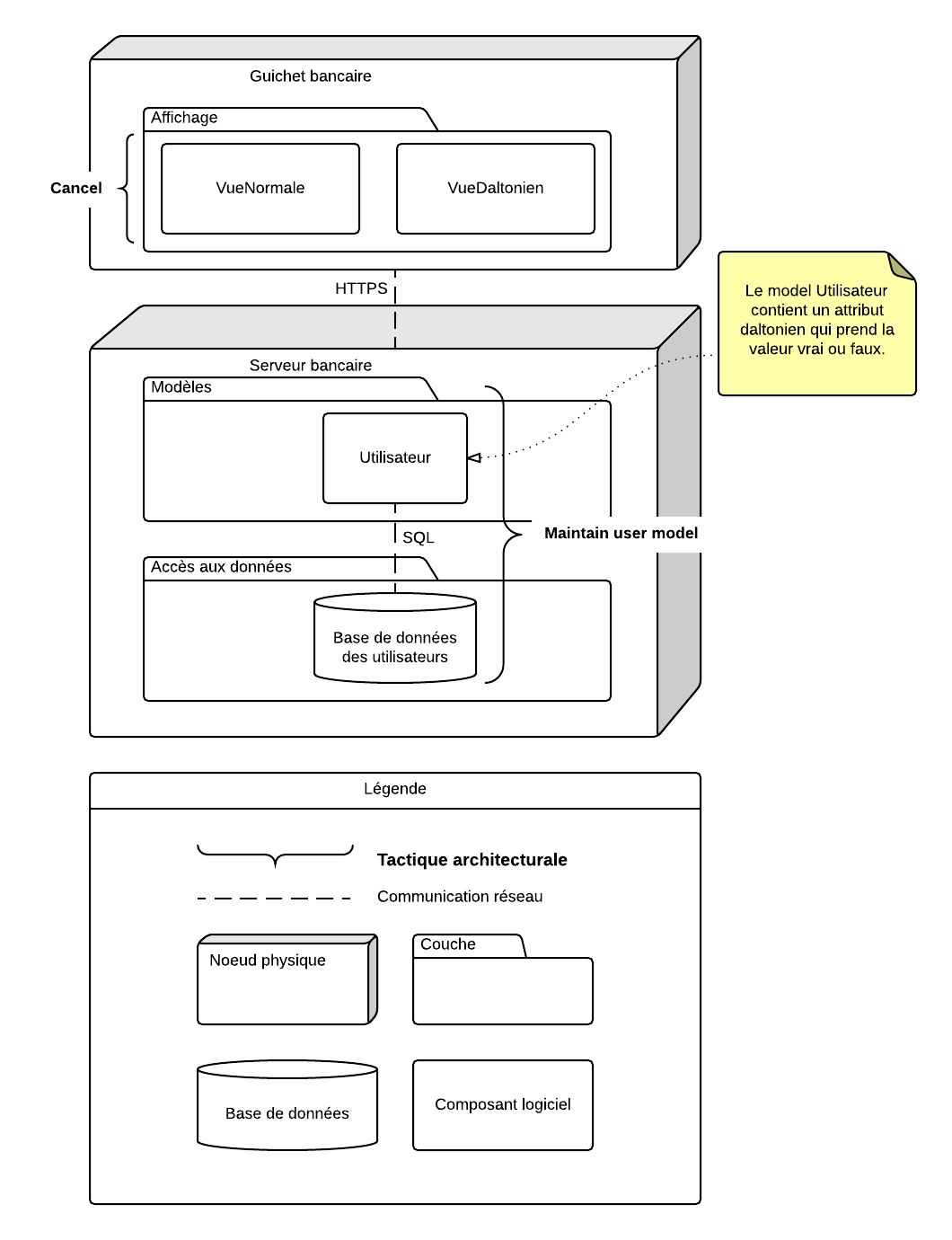
**Justification :** La banque peut sauvegarder des réglages par utilisateur spécifiant si celui-ci est daltonien. Suite à cela, le système s’adapte et change l’interface utilisateur à chaque fois qu’il utilise sa carte. Cela évite à l’utilisateur d’avoir à aller chercher le bouton pour faire le changement à chaque session.

# <Utilisabilité> – Tactique 2

**Description :** *Cancel* – Permettre à l’utilisateur de revenir au mode normal après avoir activé le mode daltonien.

**Justification :** L’utilisateur peut avoir appuyé sur le bouton de changement au monde daltonien par mégarde. Il faut donc lui permettre de revenir au mode normal. Deux options sont offertes : une confirmation est offerte lors de

# <Utilisabilité> – Vue architecturale



La vue architecturale présente l’implémentation d’une classe pour la vue adaptée aux daltoniens dans la couche de présentation et les données utilisateurs sauvegardées dans un modèle et une base de données. Le focus est mis sur ces deux aspects, les autres couches (communication, etc.) sont donc omises.

## Éléments du diagramme

|  |  |
| --- | --- |
| **Élément** | **Description** |
| Guichet | Noeud physique représentant la machine du guichet automatique bancaire. |
| Guichet / Affichage | Couche logicielle contenant les classes de présentation du guichet automatique. |
| Guichet / Affichage / VueNormale | Classe logicielle correspondant à la vue avec les couleurs normales de la banque. |
| Guichet / Affichage / VueDaltonien | Classe logicielle correspondant à la vue avec les couleurs adaptées aux daltoniens |
| Lien Guichet -> Serveur | Le guichet communique via HTTPS avec le serveur bancaire. Les détails de communications sont omis de cette vue puisque les tactiques d’utilisabilité n’y touchent pas. Pour le scénario courant, l’information importante communiquée entre le guichet et le serveur est la bascule entre le mode normal et daltonien. |
| Serveur | Serveur bancaire contenant toutes les données et le traitement relatif aux transactions bancaires. |
| Serveur / Modèles | Couche logicielle contenant les classes qui représentent des données au niveau logiciel. |
| Serveur / Modèles / Utilisateurs | Classe logicielle représentant le modèle de données d’un utilisateur. Contient plusieurs informations. Pour le scénario courant, l’information intéressante est le booléen informant si l’utilisateur est daltonien ou pas. |
| Lien Utilisateur -> Base de données utilisateurs | Lien entre le modèle utilisateur et la base de données. Le modèle obtient ses données de la base de données et les met à jour à cet endroit. Les détails d’implémentation sont omis puisque ce n’est pas le but de la vue. |
| Serveur / Accès aux données | Couche logicielle représentant l’accès aux données bancaires, utilisateurs et autres. |
| Serveur / Accès aux données / BD utilisateurs | Base de données contenant toutes les données des utilisateurs de la banque. Les détails techniques sont omis pour la simplicité du diagramme. |
| Cancel | Tactique architecturale utilisée dans la couche d’affichage / présentation. Permet d’annuler la bascule vers le mode daltonien à l’aide d’une confirmation et d’un bouton pour retourner au mode normal. |
| Maintain user model | Tactique architecturale utilisée dans les modèles de données utilisateurs. Garde l’information sur le daltonisme de l’utilisateur pour permettre de basculer dans le bon mode dès la connexion au guichet. |

La tactique *Cancel* s’applique strictement à la couche d’affichage du guichet, permettant de basculer le mode d’affichage en offrant un bouton pour retourner au mode normal et un bouton annuler lors de la bascule

La tactique *Maintain user model* s’applique à la couche des modèles de données. Les données conservées incluent un booléen pour savoir si l’utilisateur est daltonien et permettre de configurer l’interface en conséquence.

Ces deux tactiques répondent au scénario établi car ensemble, elles permettent de rendre l’utilisation du système plus agréable pour les utilisateurs daltoniens en permettant de garder en mémoire le choix d’interface tout en permettant aussi de revenir au mode normal pour un utilisateur qui aurait appuyé sur le bouton par erreur.

Groupe 05

# < Usabilité > – Justification (en classe)

Lors d’un retrait effectué par un client, on voudrait offrir la possibilité au client de recevoir une version électronique du reçu. À la fin de chaque retrait, le système propose au client d’envoyer une copie du reçu par SMS ou par courriel. De cette façon, il pourrait mieux garder ses archives dans le «cloud».

# <Usabilité> – Scénario 1 (en classe)

Source: Utilisateur

Stimuli: Minimiser les impacts des erreurs

Artéfact: GAB

Environnement: Runtime

Réponse: On propose une nouvelle fonctionnalité au client

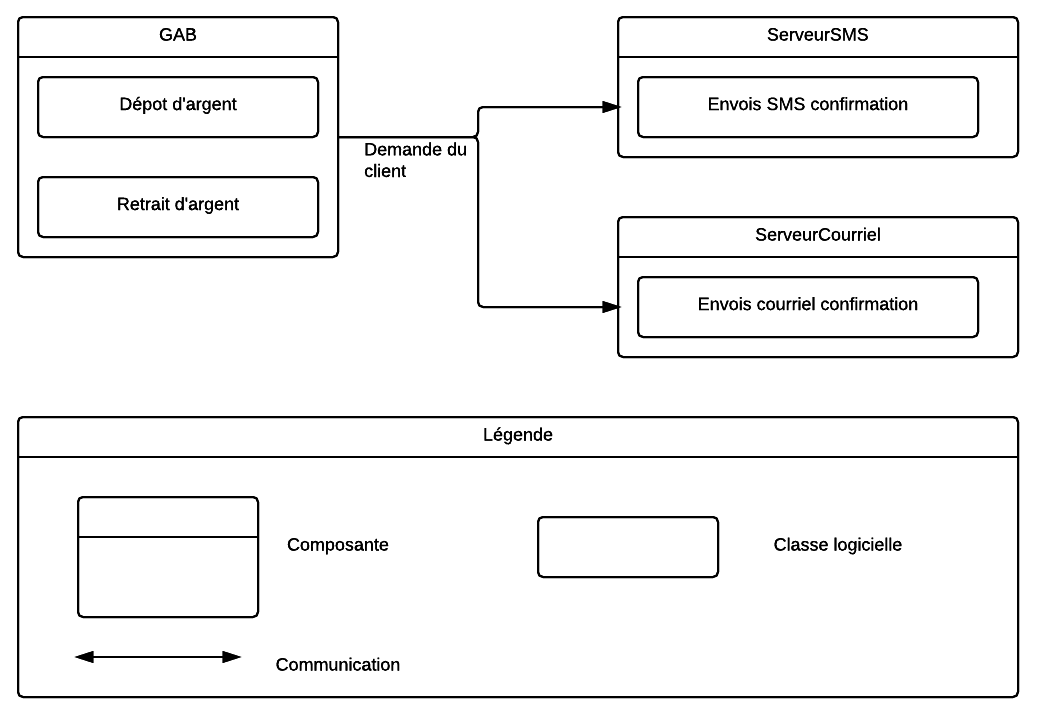
Mesure de la réponse: On augmente la satisfaction du client

# <Usabilité> – Tactique 1 (en classe)

Description: Maintain system model

Justification: On utilise cette tactique pour pouvoir donner une réponse appropriée à la requête de l’utilisateur. Dans ce cas, le client veut recevoir son reçu en version électronique. Le système lui donnera donc le choix de le recevoir par courriel ou par SMS.

# <Usabilité> – Vue Architecturale (en classe)



* Texte de description du diagramme

Ce diagramme représente un client qui a le choix de recevoir une copie de son reçu par SMS ou par courriel suite à une transaction.

* Table de description des éléments du diagramme

|  |  |
| --- | --- |
| **Éléments** | **Description** |
| GAB | Guichet automatique bancaire que le client utilise pour effectuer un retrait |
| ServeurSMS | Serveur de la banque qui gère la manipulation de SMS. |
| ServeurSMS | Serveur de la banque qui gère la manipulation de courriel. |
| Dépot d’argent | Classe logicielle qui sert à effectuer un dépot d’argent. |
| Retrait d’argent | Classe logicielle qui sert à effectuer un retrait d’argent. |
| Envoie SMS confirmation | Classe logicielle qui permet d’effectuer l’envoie du SMS de confirmation. |
| Envoie courriel confirmation | Classe logicielle qui permet d’effectuer l’envoie du courriel de confirmation. |

Groupe 08

# < Usabilité > – Justification (en classe)

Le client peut utiliser un ATM pour faire un transfert d’argent vers le compte d’une autre personne, ou payer une facture d’une compagnie avec laquelle il fait affaire.

Il doit souvent faire ces transactions avec les mêmes comptes d’autres personnes ou avec les mêmes compagnies avec lesquelles il fait affaire.

Il serait souhaitable d’accélérer cette opération en ne saisissant que la première fois au guichet (ou au comptoir ou sur le portail Web de la banque) les informations de la compagnie qui a émis la facture ou le compte dans lequel on veut transférer l’argent.

Les prochaines fois, on aurait qu’à choisir le compte dans lequel on veut faire le virement ou la compagnie à payer dans une liste.

# <Usabilité> – Scénario 1 (en classe)

Source: Le client de la banque.

Stimuli: Vouloir payer une facture ou transférer des fonds dans le compte d’une autre personne.

Artéfact: Le guichet ATM.

Environnement: Runtime

Réponse: Le paiement de facture ou le virement a été fait, en choisissant la personne ou compagnie dans une liste construite à partir de l’historique de transactions.

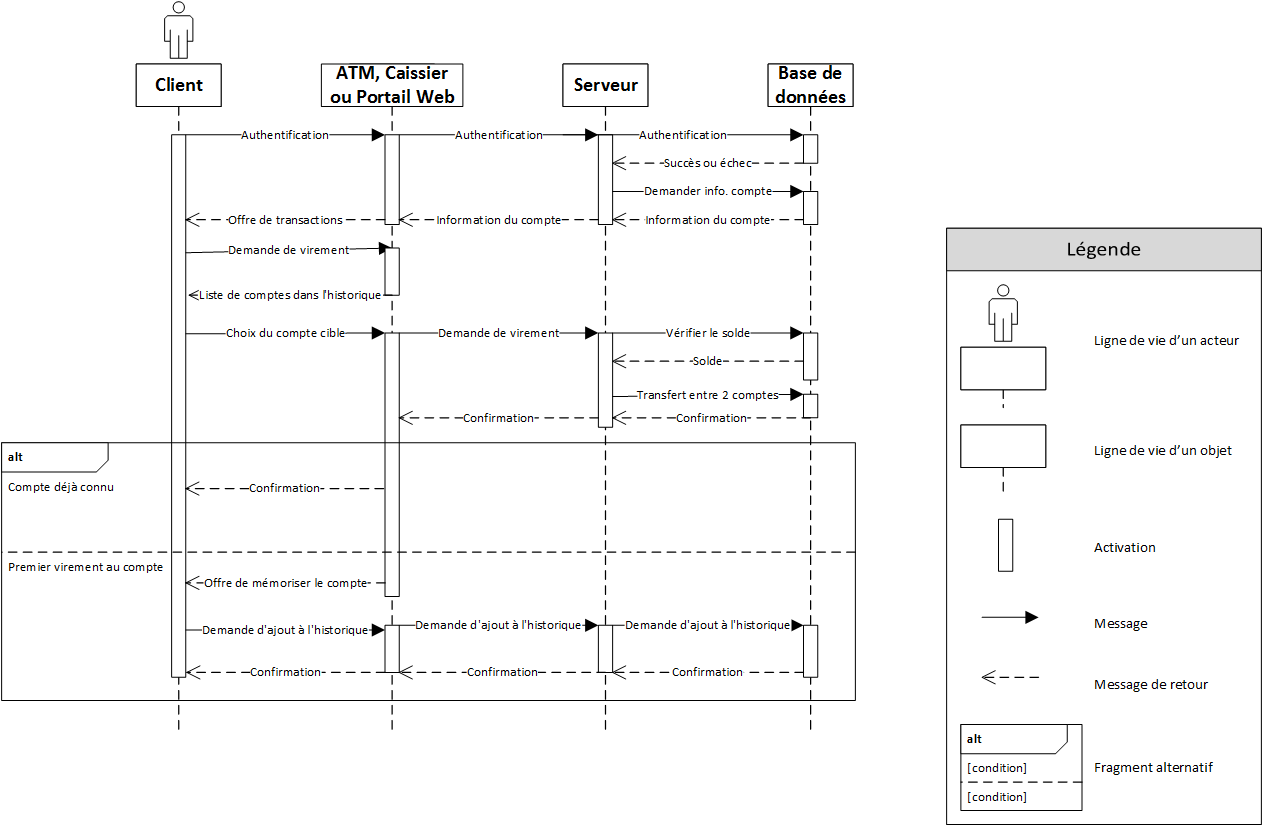
Mesure de la réponse: Le temps nécessaires pour effectuer le virement ou le paiement de facture, pour une personne ou une compagnie avec laquelle on a déjà fait des transactions. La probabilité faible qu’on se trompe en entrant les informations de la compagnie ou person­ne, étant donné que le client n’a à les saisir qu’une seule fois.

# <Usabilité> – Tactique 1 (en classe)

Description: Support System Initiative – Maintain task / user model

Justification: En se basant sur les tâches effectuées par le passé (virements et paiement de factures), on peut prédire que de nouvelle tâches ont de fortes probabilités d’être faites (avec les mêmes compagnies à payer ou compte dans lesquels faire un virement). En connaissant les habitudes de l’utilisateur, le système peut donc afficher une liste de ces choix possibles et ainsi améliorer l’utilisabilité en prenant cette initiative.

# <Usabilité> – Vue Architecturale (en classe)



**Description des éléments du diagramme**

|  |  |
| --- | --- |
| Élément | Description |
| Client | Personne qui possède un compte avec la banque à laquelle appartient l’ATM et qui souhaite faire un virement vers le compte d’un autre client (de la même banque dans ce scénario). Il peut ou non avoir déjà fait des virements par le passé vers ce compte ou d’autres. Il doit avoir en sa possession sa carte bancaire et connaitre son NIP. |
| ATM | Le guichet automatique est installé dans une succursale de la banque et possède un accès à Internet |
| Caissier | Le caissier utilise un terminal et demande lui-même verbalement si le client veut mémoriser ou réutiliser un compte vers lequel faire un virement. |
| Portail Web | Le portail Web permet de faire des transactions à partir d’un navigateur Web. Il set distinct du serveur de transactions. |
| Serveur (de transactions) | Serveur central de la banque qui permet de s’authentifier et d’effectuer des transactions entre un ou plusieurs comptes (de la même banque dans ce scénario) de manière sécuritaire. Il accepte des requêtes de mémoriser l’historique des comptes vers lesquels le client a fait des virements. |
| Base de données | Serveur qui exécute un moteur de base de données qui contient notamment des informations sur les clients, leurs comptes et ceux vers lesquels il ont fait des virements. Ces données sont répliquées à plusieurs emplacements sécurisés et font l’objet de processus stricts de validation. |
| Information du compte | Information stockée dans la base de données du serveur central de transactions et retournée à l’ATM, au terminal du caissier ou au portail Web suite à une authentification réussie. Elle comporte des informations sur le client, ses comptes et une liste des personnes avec lesquels il a mémorisé qu’il a déjà fait des virement. |

**Description du diagramme**

Ce diagramme de séquence UML représente l’interaction entre un client d’une banque qui veut faire un virement vers le compte d’un autre client de la même banque, par l’entremise d’un ATM. Le portail Web, et le caissier sont représentés parce qu’ils offrent au client, comme l’ATM, l’option de mémoriser le compte de destination, pour que celui-ci puisse être choisi parmi une liste, lors d’un prochain virement. Les étapes de la transaction sont l’authentification, le choix d’un virement comme transaction, le choix d’un compte cible, possiblement parmi une liste de suggestions, et finalement la demande au client s’il veut mémoriser ce compte pour pouvoir faire de futurs virements.

**Relation entre les éléments et la tactique**

Le serveur, suite à une authentification réussie, retourne l’information du client qui utilise l’ATM ou communique avec le portail Web ou le caissier. Cette information inclut l’historique des virements passés, dans le but de fournir de l’assistance à l’utilisateur dans sa tâche de possiblement réaliser un nouveau virement, étant donné qu’on peut s’attendre à ce que des virements se répètent dans les mêmes comptes.

**Comment la tactique répond à chacun des scénarios de qualité**

La tactique qui consiste à garder un modèle de la tâche et de l’utilisateur répond au scénario d’usabilité en permettant au client de faire des virements récurrents plus rapidement et sans la frustration de devoir ressaisir les informations des comptes cibles en courant la chance de se tromper. En ayant le choix de ne pas mémoriser les comptes, il peut éviter de surcharger sa liste de choix suggérés, ce qui lui évite la frustration et la perte de temps à parcourir plus tard une longue liste de comptes suggérés

Equipe 10

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

La banque désire faciliter l’utilisation du GAB par les clients, par exemple pour accélérer le traitement des dépôts. Ceci pour permettre aux clients de passer moins de temps à la banque, et donner à la banque la possibilité de servir plus de clients.

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

**Source**: Utilisateur du GAB

**Stimuli**: Utilisateur qui veut faire des dépôts plus efficaces, rapides

**Artéfact**: Interface utilisateur du GAB

**Environnement**: Runtime

**Réponse**: Le GAB anticipe les besoins de dépôts de l’utilisateur selon ses habitudes, propose des choix de dépôts rapides

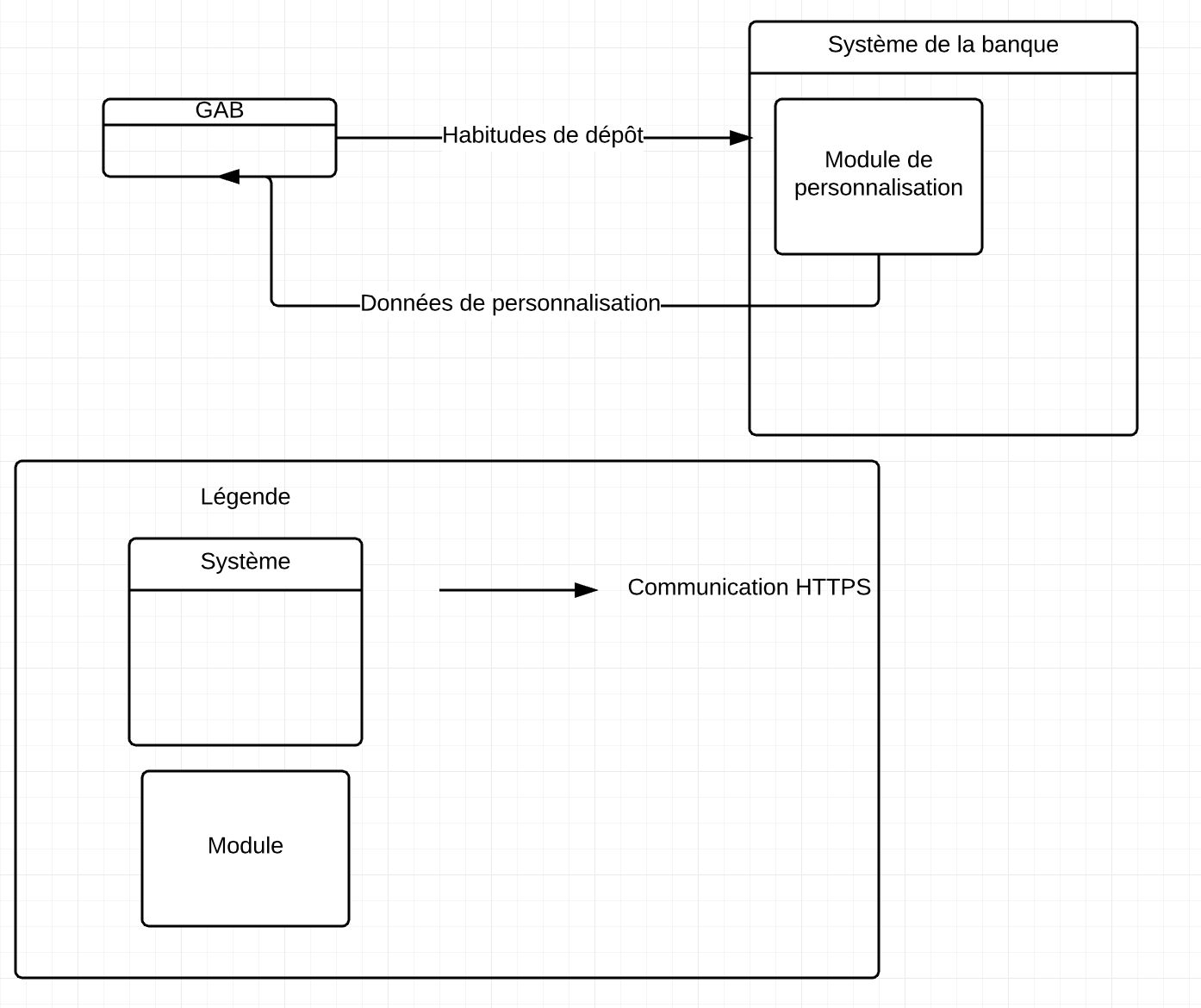
**Mesure de la réponse**: Le temps gagné pour les opérations de dépôts, satisfaction de l’utilisateur, nombre de dépôts dans une journée

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

**Description**: *Maintain User Model, Maintain System Model*

**Justification**: L’utilisateur voulant faire un dépôt rapide va recevoir une réponse du système avec une interface personnalisée qui affiche des montants de dépôts selon ses habitudes.

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



Le GAB, lors d’une opération de dépôts, communique au système de la banque pour sauvegarder les habitudes de dépôt de l’utilisateur. Ce faisant, le GAB reçoit aussi les informations des habitudes de l’usager pour afficher l’interface de dépôt à l’usager pour la transaction courante.

**Table de description des éléments du diagramme**

|  |  |
| --- | --- |
| **Élément** | **Description** |
| GAB | Solution logicielle implémentée dans le guichet automatique bancaire |
| Communication HTTPS | Canal de communication pour les informations de personnalisation des clients |
| Système de la banque | Serveurs externes avec les différents modules qui gèrent l’utilisation du GAB |
| Module de personnalisation | Banque de données et interface pour les informations de personnalisation des clients |

Le système garde un modèle de lui­même pour servir au client des interfaces qui permettent le dépôt rapide. Il garde aussi les informations de personnalisation des clients pour voir leurs habitudes de dépôts et les injecter dans l’interface.

Avec un dépôt rapide, la banque peut servir plus de clients en augmentant la facilité d’utilisation de l’opération de dépôt. Cette opération est probablement celle qui prends le plus de temps aux utilisateurs, et en améliorer le processus sauve du temps et à la banque et à l’utilisateur.