**Réalisation d’un agrégateur de moyen de paiement**

**CACHIER DE CHARGE DETAILLÉES**

[1 Objectif 3](#_Toc47337431)

[1.1 Objectif général 3](#_Toc47337432)

[1.2 Objectif spécifiques 4](#_Toc47337433)

[2 Analyse des besoins 4](#_Toc47337434)

[2.1 Spécifications fonctionnels 4](#_Toc47337435)

[2.2 Les besoins non fonctionnels 5](#_Toc47337436)

[3 Conception 5](#_Toc47337437)

[3.1 Modélisation fonctionnelle 6](#_Toc47337438)

[3.1.1 Identification des acteurs du système 6](#_Toc47337439)

[3.1.2 Diagramme de cas d’utilisation 6](#_Toc47337440)

[3.1.3 Diagramme de classe 7](#_Toc47337441)

[3.2 Modélisation dynamique 7](#_Toc47337442)

[3.2.1 Diagramme de séquence 7](#_Toc47337443)

[3.2.2 Diagramme d’activité 7](#_Toc47337444)

[3.3 Spécifications techniques 7](#_Toc47337445)

[3.3.1 Serveur cloud dédié ou mutualisé 8](#_Toc47337446)

[3.3.2 Technologies recommandées 8](#_Toc47337447)

[3.3.3 Sécurité de l’application 9](#_Toc47337448)

[3.3.4 Ergonomie 10](#_Toc47337449)

Sigles et abréviations

|  |  |
| --- | --- |
| Termes | Définitions |
| API | *Application Programming Interface* |

# Objectif

## Objectif général

L’objectif principal de ce travail est de mettre en place une solution d’agrégation de divers services de paiement mobile disponibles au Burkina Faso qui, une fois intégrée dans une plateforme/application de collecte d’argent ou de e-commerce, offre nativement la possibilité d’utiliser les moyens de paiement existant au Burkina Faso pour régler des paiements de manière électronique. L’agrégation effectuée permet également au service de collecte de procéder à des versements au contribuable (remboursement, rollback d’une transaction, paiement d’une prestation à un contribuable). Il s’agit d’offrir :

* Au client final (ici le contributeur payeur) une expérience de paiement homogène et simple ;
* Aux développeurs une intégration facilitée ;
* Au service de collecte, une gestion aisée de la relation avec le contributeur au moyen d’outils de suivi de transactions ;
* Une sécurité et une gestion de la fraude accrue, venant compléter et renforcer les mesures préalablement mises en place par les solutions de paiement agréées.

## Objectif spécifiques

Les objectifs spécifiques de cette mission sont les suivants

* Acquérir un serveur Cloud public dédié ou mutualisé pour l’hébergement de la plateforme;
* Mettre en ligne la solution sur un service Cloud public sécurisé et performant;
* Mettre la solution en phase de transition pour la Pré-production;
* Lancer la solution en production de manière graduelle et de préférence selon la méthodique éprouvée de gestion agile SCRUM des projets de développement et de mise en œuvre logiciel ;
* Assurer la maintenance l’évolution de la solution

Assurer un support permanent au développeurs qui vont intégrer la solution sur leurs applications. Ce support consistera principalement en: o

* L’assistance au développeurs pour résoudre les problèmes et difficultés fonctionnelles ou techniques qu’ils rencontrent ;
* La mise en place d’un dispositif de gestion des demandes d’interventions couramment appelé HelpDesk :
* Accès à un site WEB Helpdesk pour la remontée des incidents 24h/24

# Analyse des besoins

L’analyse consiste à l’aboutissement de l’élaboration d’une solution technique à partir de l’étude des besoins. C’est la première phase du cycle de développement d’un logiciel. Elle sert à identifier les acteurs du système et leur associer chacun l’ensemble des actions avec lesquelles il intervient dans l’objectif de donner un résultat optimal et satisfaisant au client. On distingue deux types de besoins : les besoins fonctionnels et non fonctionnels.

## Spécifications fonctionnels

Les besoins fonctionnels répondent aux points précis du cahier de charges et sont donc requis par le client. Ils constituent le besoin primaire du client et définissent une fois résolus l’opérationnalité du système. Ainsi l’agrégateur de moyen de paiements doit permettre :

* Au financier de :
* Consulter la liste des clients ;
* Consulter la liste des reversements ;
* Consulter la liste des transactions ;
* Consulter le tableau de bord du marchand ;
* Au développeur de :
  + Consulter les logs ou rapports du système applicatif ;
  + Ajouter un webhook ou une URL de rappel de données ;
  + Consulter la liste des webhooks ;
  + Consulter les diverses clés API mises à disposition ;
  + Copier une clé API ;
* Au marchand ou à l’administrateur de :
  + Initier une transaction mobile money ;
  + Rembourser une transaction ;
  + Configurer les reversements ; les reversements d’avoirs peuvent être périodique, spontané ou par palier.
  + Ajouter un compte dans le cadre des reversements ;

Il est à noter que le compte ajouter par ce dernier dans le cadre des reversements peut aussi bien être un compte mobile money qu’un compte bancaire.

* + Supprimer un compte ajouté dans le cadre du reversement d’avoirs ;

Cette action ne peut toutefois être exécutée que s’il existe rattaché au compte marchand un autre compte ajouté dans le cadre des reversements.

* + Demander l’activation de son compte ; l’activation d’un compte marchand implique le téléversement par ce dernier des pièces que sont : le Relevé d’Identité Bancaire (RIB), l’Identité Fiscale Unique (IFU), le Régistre du Commerce et du Crédit Immobilier (RCCM) et le numéro d’une quelconque pièce d’identité (passeport ou carte d’identité).
  + Ajouter un business ;
  + Supprimer un business ;
  + Inviter un utilisateur sur la plateforme ;

Le marchand ou l’administrateur du compte peut choisir d’inviter sur ce dernier un utilisateur. A ce dernier, peuvent être attribués les rôles qui suivent : financier, développeur ou administrateur.

* + Supprimer un utilisateur ;
  + Générer un lien de paiement ;
  + Supprimer un lien de paiement.

## Les besoins non fonctionnels

Ces besoins sont soit des besoins optionnels, soit des besoins/contraintes liés à l’implémentation. Ainsi, il faudra que l’application web soit sécurisée, performante, dotée d’une bonne expérience utilisateur et compatible à la plupart des navigateurs.

# Conception

Cette phase permettra de définir de manière non ambigüe, en utilisant un langage de modélisation, le fonctionnement futur du système, afin de faciliter sa réalisation. La modélisation de notre système s’articulera autour des diagrammes qui suivent :

* Le diagramme de cas d’utilisation ;
* Le diagramme de classe ;
* Le diagramme d’objet ;
* Le diagramme de séquence ;
* Le diagramme d’activité. L’on distingue les différents types de modélisation que sont : la modélisation fonctionnelle, la modélisation statique et la modélisation dynamique.

## Modélisation fonctionnelle

Comme tout bon système d’informations logiciel au cœur de la performance d’une structure, la solution finale doit permettre l’automatisation de l’ensemble des processus identifiés par les besoins ; d’acquisition des données, leur stockage sécurisée, optimisée et disponible jusqu’à l’édition des rapports de gestion statistiques/financiers et des tableaux de bord décisionnels pour le management.

Seront présentés dans cette partie les différents acteurs du système ainsi que les différentes interactions réalisables avec ce dernier. En effet les diagrammes fonctionnels rassemblent : le diagramme des cas d’utilisation (use-cases ou Use Case Diagram), le diagramme de contexte statique, le diagramme de contexte dynamique. Nous présenterons ici que le diagramme de cas d’utilisation.

### Identification des acteurs du système

En UML, on entend par acteur toute idéalisation d’un rôle joué par une personne externe, un processus ou une entité qui interagit avec le système. Il existe deux types d’acteurs, à savoir : les acteurs principaux qui utilisent directement les fonctionnalités du système et les acteurs secondaires qui contribuent à la réalisation d’un cas d’utilisation. Dans le cadre de notre système nous avons retenu :

* Comme acteurs principaux :
  + Le financier;
  + Le développeur ;
  + Le marchand ou l’administrateur ;
* Comme acteur secondaire :
  + L’administrateur du système.

### Diagramme de cas d’utilisation

### 

Le diagramme de cas d’utilisation présente le comportement de notre système, d’un package ou d’un composant tel qu’un utilisateur extérieur le voit. Il scinde donc la fonctionnalité du système en unité cohérente : cas d’utilisation, qui permet d’exprimer les besoins des utilisateurs de notre système.

Diagramme de cas d’utilisation ici

### Diagramme de classe

Ce diagramme permet de représenter une vue statique du système d’informations. Statique, car l’on n’évoquera point ici les stimuli qui font réagir le système mais plutôt des relations entre les classes, des services rendus et utilisés par chacune d’elles et de l’articulation de l’ensemble.

Diagramme de classe ici

## Modélisation dynamique

Les diagrammes dynamiques permettent de modéliser le comportement dynamique du système et d’indiquer comment ses objets interagissent au moment de l’exécution.

### Diagramme de séquence

Le diagramme de séquence qui représente la séquence met à l’honneur la succession de messages entre objets au cours d’une interaction. Il comprend un groupe d’objets, représentés par des lignes de vie, et les messages que ces objets échangent lors de l’interaction et représente la succession chronologique des opérations réalisées par un acteur.

Diagramme de séquence ici

### Diagramme d’activité

Le diagramme d’activité permet de modéliser le comportement du système, dont la séquence des actions et leurs conditions d’exécution. Les actions sont les unités de base du comportement du système. Un diagramme d’activité permet de grouper et de dissocier des actions. Si une action peut être divisée en plusieurs actions en séquence, vous pouvez créer une activité les représentant.

## Spécifications techniques

La réalisation d’un système demande une hiérarchisation des idées et outils liés. Nous procéderons à un inventaire des outils liés à la mise en place de l’architecture sécurisée de notre système et nous aborderons la mise en place proprement dite du système.

### Serveur cloud dédié ou mutualisé

Toute la future plateforme de l’agrégateur de moyen de paiement, de l’API à l’application web devrait être hébergée sur un espace (serveur dédié ou serveur mutualisé) Cloud public acquis auprès d’un fournisseur qui garantit un niveau de SLA (Service Level Agreement) hautement disponible. L’espace (serveur dédié ou serveur partagé) à acquérir doit correspondre à un abonnement d’une année et doit avoir les caractéristiques minimales suivantes :

* CPU : 2\*2.4 GHZ
* RAM : 8 Go
* Disque dur : 500 Go
* Bande passante : 500 Mbps

Selon les technologies de réalisation et de déploiement de la solution, soit Microsoft (ASP.NET) ou Open source autour de Linux (notamment, JAVA, PHP ou PHYTON), des coûts de services supplémentaires doivent être pris en compte en termes de système d’exploitation, de serveur d’applications, de serveur de base de données, etc.

Nous suggérons des fournisseurs de renommé mondiale (Microsoft Azure, OVH, LWS, etc.) pour leur expérience éprouvée en termes de qualité de SLA, la sécurité des infrastructures, la bande passante, le coût et la réactivité face aux incidents.

Il s’agit de fournir les prestations suivantes :

* Abonnement à un service de fourniture d’un serveur cloud public ;
* Installation de l’ensemble de composants logiciels nécessaires au déploiement ;
* Tests de bon fonctionnement du matériel et des composants logiciels ;
* Transfert de compétence sur l’exploitation technique des composants logiciels ;
* Fourniture des documents de spécifications techniques et des configurations des composants logiciels.

### Technologies recommandées

* Système de gestion de bases de données

Un système de gestion de base de données (SGBD) exigé avec une licence valide ou un contrat de souscription annuel dans le cas d’une base de données open source. Nous préconisons : Ms SQL Server, MySQL Enterprise ou PostgreSQL Enterprise DB.

* Outils de développement

Langages de programmation : HTML5, PHP, Python, XML, CSS3, Bootstrap, ASP.NET JavaScript, Java. L’usage d’un Framework abouti Java / PHP ou Python est fortement recommandé pour donner de la célérité.

Un environnement purement Windows pourrait être envisagé si les coûts sont accessibles : serveur Cloud Microsoft Azure / Windows server / Environnement de développement ASPT.NET/ Serveur d’applications IIS.

* Serveur d’applications

Un serveur d’applications commercial ou open-source éprouvé, robuste et sécurité doit être utilisé pour l’hébergement des applicatifs. Il s’agit notamment de Glassfish, Joss, Wilfly, Tomcat, Apache, IIS

### Sécurité de l’application

Le système traite des informations sensibles propres aux divers utilisateurs. Du fait de la transition des données et de l’interconnexion des différentes applications un certain nombre de mesures sécuritaires doivent être prises afin de réduire autant que possible les failles de sécurité. Nous classons ci-dessous les mesures que nous avons adoptées en fonction des principes de sécurité informatique :

* La confidentialité

La confidentialité est le fait de s’assurer qu’une information est accessible uniquement par les entités qui ont le droit d’accéder à celle-ci. Le système doit intègrer cet aspect de la sécurité à deux niveaux :

* Dans un premier temps nous devons installer un certificat ssl de 2048 bits qui nous permettra de sécuriser toutes les communications utilisant le protocole tcp/ip;
* Dans un second temps un système de gestion de rôle et de droit intégré nous permet de respecter la confidentialité des données dans chaque sous système. En effet, les utilisateurs appartiennent à des groupes d’utilisateurs donnés et pour chaque groupe seulement un ensemble d’actions bien définies sont possibles.
* L’intégrité

L’intégrité s’assure que la donnée reste toujours intègre c’est-à-dire qu’elle n’a pas été modifiée par un tiers non autorisé. Ce principe de sécurité est pris en compte par la mise en place d’un certificat ssl sur le serveur d’applications.

* La disponibilité

La disponibilité est le fait de s’assurer que l’information soit toujours disponible peu importe le moment choisi. Elle sera assurée en partie par l’infrastructure mise en place comme des serveurs de redondance mais aussi par la virtualisation des ressources et des applications qui seront déployées. Aussi une gestion de la montée en charge sera implémentée.

* L’authenticité

Au sein d’un système d’information, il est important de vérifier l’authenticité de chaque ressource. Cela est possible grâce au mécanisme d’authentification, qui permet de prouver l’identité d’une personne via le processus d’identification. Cet aspect sera mis en place par une double authentification. Dans un premier temps avant de faire appel à une quelconque URI, il faudrait être authentifié avec un couple d’identifiant comme service tiers de confiance. Donc nos applications clientes s’authentifient avant de demander un quelconque service au backend.

* Non-répudiation

La non-répudiation se base sur un principe simple : une entité ne peut nier son implication dans une action à laquelle il a participé. Dans notre cas la non-répudiation peut être atteinte seulement en utilisant la technologie du certificat numérique avec un chiffrement asymétrique.

* Imputabilité

Il s’agit des techniques mises en œuvre pour assurer la traçabilité des actions d’un individu sur un système. Nous allons mettre en place cette traçabilité en journalisant toutes les actions menées sur les objets du système.

### Ergonomie

L’ergonomie Web devra être responsive design, c’est-à-dire que l’interface utilisateur est flexible et dynamiquement adaptable au PC de bureau, Pc portable, Tablette, smartphone.

Elle doit favoriser la lisibilité, l’accessibilité des informations, leur réutilisation et le respect de la norme ISO 9241 :

* L’application back-end doit avoir une interface graphique conviviale et permettre une navigation simple entre les menus ;
* Il doit être simple et intuitif dans l’utilisation permettant ainsi aux utilisateurs non-initiés de s’adapter facilement ;
* A tout moment, l’utilisateur doit disposer d’un moyen d’impression et/ou d’import-export (au besoin) intégré à l’outil ;
* A tout moment, l’utilisateur doit disposer d’une aide en info bulle ou autre sur l’utilisation des fonctionnalités, actions de l’interface concernée ;
* Un système d’aide en ligne devra être disponible ;
* L’impression d’écran devra toujours être disponible ;
* La navigation contextuelle ;
* Les recherches multicritères à base de filtres avancés ;
* Les possibilités de tri ;
* La différenciation par la couleur des différents types de champs (modifiable, lecture seule, obligatoire/non obligatoire) ;
* La possibilité d’activation de saisie assistée ;