# SPECIFICATION TECHNIQUE DE BESOIN

**Version :** 1.0

**Date :** 11/01/2017

**Rédigé par :** L'équipe SmartLogger

**Relu par :** L'équipe SmartLogger

**Approuvé par :** ---

*Signature du superviseur :*

**Objectif :** Ce document est destiné à traduire les besoins des utilisateurs du logiciel et à établir une référence pour sa validation. Il doit être élaboré en coopération avec le demandeur puis approuvé par ce dernier. Son but est de recenser les principales exigences que l'équipe de développement s'engage à satisfaire dans le cadre du projet.

# HISTORIQUE DE LA DOCUMENTATION

| Version | Date | Modifications réalisées |
|---|---|---|
| 0.1 | 03/11/16 | Création du document |
| 0.2 | 10/11/16 | Augmentation du contenu et divers correctifs |
| 0.3 | 29/11/16 | Simplification de la schématisation des cas d'utilisation |
| 1.0 | 11/01/17 | Correctifs et restructuration du document |

## 1. Objet :

Le but du projet est de créer un système, permettant d'alerter l'utilisateur sur des données en provenance d'applicatifs défectueux dans l'optique de faciliter leurs correctifs. Une fois produit, ce système sera utilisé par l'entreprise cliente à leurs propres fins.

L'objectif est donc de mettre en place un système d'alarme qui disposerait de ses propres capacités d'apprentissage. Sa tâche principale sera de prioriser les différentes données à traiter selon leur niveau de criticité et d'adopter une stratégie de prévention adaptée.
Sur le long terme, ce dernier devra améliorer son analyse en exploitant ses résultats précédents afin de constamment améliorer le taux de confiance des alertes émises. L'enjeu sera donc de créer un système combinant facultés d'analyse et de prédiction.

Dans cette optique, l'entreprise client nous a recommandé l'emploi de certaines technologies qui, pour la plupart, sont connues et utilisées pour leurs propres projets.

Ceci inclut :

- Spring Boot : afin de réaliser la partie applicative.
- Angular 2 : pour la réalisation de l'interface graphique.
- Spark : pour la mise en place des algorithmes.

Le développement du projet entier ne repose sur aucune fonctionnalité ou logiciel existant.
De ce fait, l'équipe projet devra concevoir le fonctionnement du système dans son intégralité et en assurer la production.
Cependant, comme demandé par le client, le produit final devra respecter les contraintes suivantes :

- Il devra analyser des flux de données de type Shinken, Logstash et Apache Kafka.
- Il alertera les opérateurs en employant des moyens de communication utilisés par l'entreprise, tels que l'application Slack, l'envoi de mails ou par notification SMS.
- Il pourra s'adapter à des flux de données ou méthodes d'alerte non prédéfinies qui seront implantées dans le système par de futurs utilisateurs.
- Enfin, il devra fonctionner sur de longues périodes de fonctionnement et, dans l'idéal, être opérationnel indéfiniment.

## 2. Documents applicables et de référence

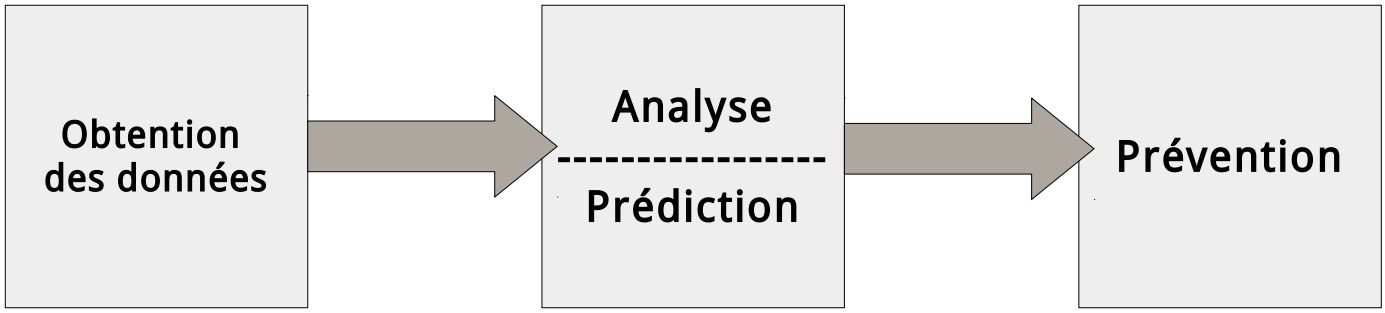
- Le document de spécification client : SmartLogger.pdf
- Le glossaire associé à la documentation : Glossaire.pdf

# 3. Exigences fonctionnelles

## 3.1. Présentation de la mission du produit logiciel

Le logiciel final, bien que scindé en plusieurs parties distinctes, sera déployé sur un serveur externe appartenant au client puisqu'il nécessite divers flux de données issus de systèmes extérieurs. Il agira en tant qu'intermédiaire entre les logs et les opérateurs de maintenance, prenant ainsi position dans ce type de procédure.

Sa fonction principale reposera sur la mécanique suivante :




- La première phase consiste en la récupération des données. En effet, vu qu'elles proviennent de sources extérieures au système, il faudra veiller à leur conformité mais également à la cohérence de leur contenu. Le but étant de protéger la machine contre des attaques externes (e.g. : injections de code) ou échantillons de données qui tronqueraient le modèle de prévision mis en place.

- La deuxième phase consiste en l'analyse de ces dites données. Celle-ci s'opère à deux niveaux :
  - Une analyse brute qui déterminera l'état de l'applicatif en fonction des données physiques présentes sur le log. Il permet de déterminer l'état d'une application à un instant donné.
  - Une analyse prédictive qui emploiera des algorithmes de Machine Learning pour déterminer l'état d'une application dans un futur proche.

  Cette analyse permettra d'identifier des défaillances dans certains logiciels. Ces défaillances seront alors classifiées puis triées selon leur niveau de criticité.

- La troisième phase exploite la classification précédente : si le résultat d'une analyse témoigne d'un état critique, le système va alors alerter les différents opérateurs concernés au moyen de canaux de communication (Slack, Mail, SMS).

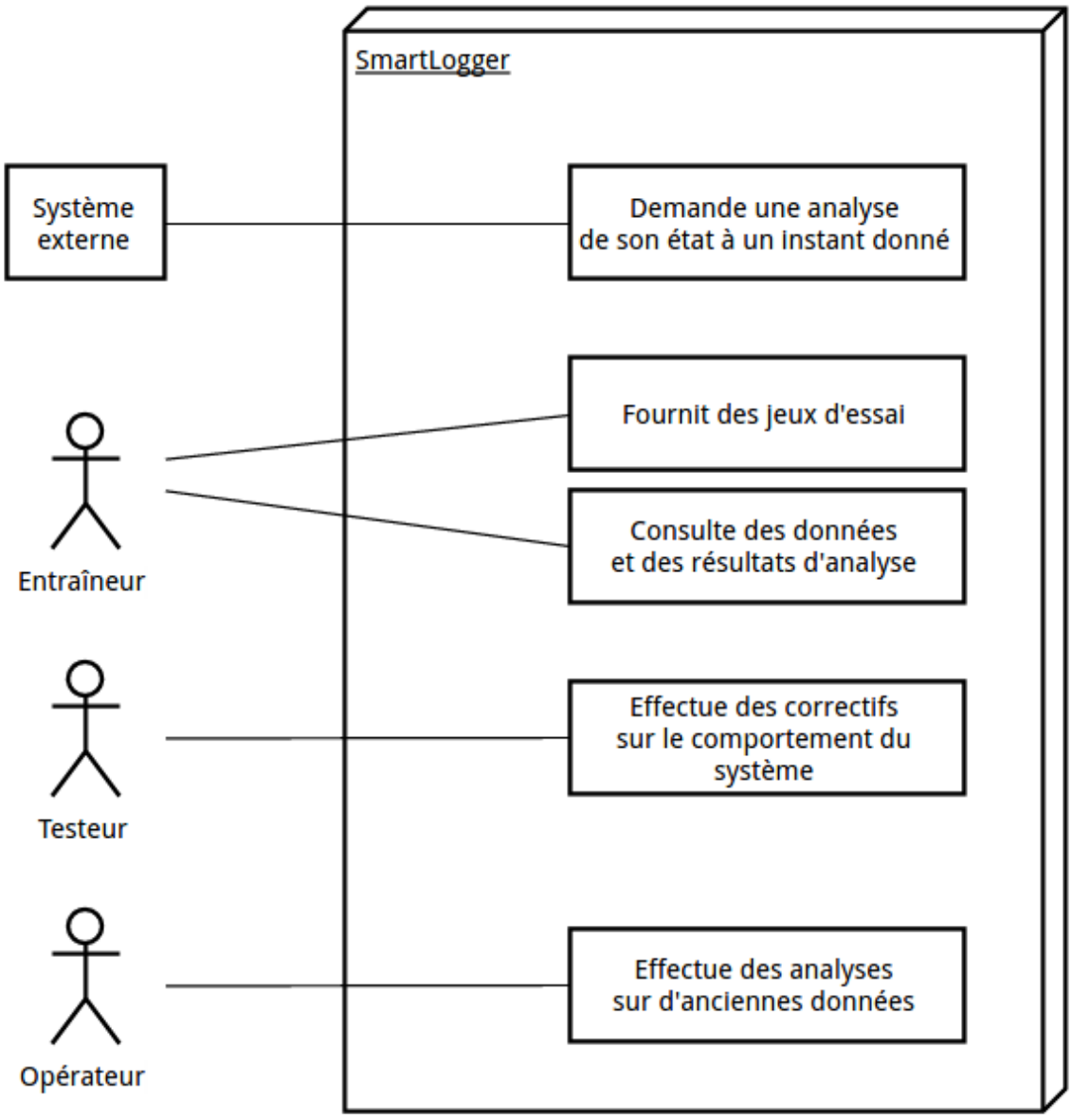
Néanmoins, pour fournir une telle capacité de prédiction au système, il va falloir fournir des jeux d'essais et offrir la possibilité d'ajuster les traitements effectués afin de perfectionner le processus d'analyse prédictive.
Le système devra donc proposer à un utilisateur, via une interface graphique, de pouvoir modifier la façon dont un échantillon de données précis doit être convenablement traité, mais également de pouvoir rajouter des échantillons supplémentaires pour augmenter la capacité prédictive du logiciel.
Ainsi, en fonction de ses tâches, le système accordera un certain niveau de permission selon le rôle à assurer, ce qui permet de partitionner la fonction de chaque utilisateur vis-à-vis du système.

De ce fait, le système distinguera 4 profils d'utilisateurs différents :

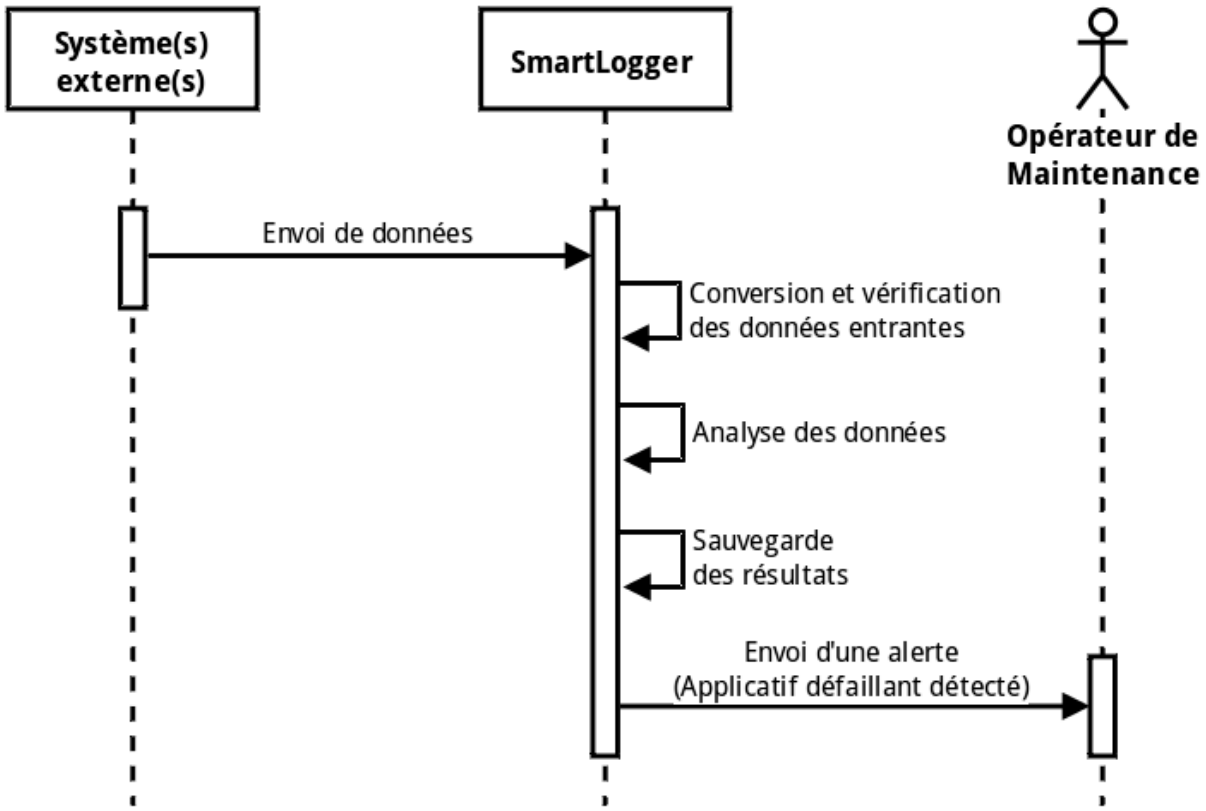
- Les **systèmes externes :**
  Les acteurs principaux de ce système, ils correspondent à des systèmes/logiciels informatique extérieurs, qui lui communiquent de nouvelles données à traiter.
- Les **entraîneurs :**
  Des utilisateurs humains qui réalisent l'apprentissage initial du système en lui fournissant des jeux d'essais.
- Les **testeurs :**
  Des entraîneurs particuliers possédant le droit supplémentaire de modifier le comportement de l'analyse. (Droit de réglage sur le système)
- Les **opérateurs :**
  Des utilisateurs polyvalents possédant des droits d'utilisation supplémentaires, représentant les employés liés à ce système.
  Ils peuvent ainsi assurer le rôle de testeur, donner des ordres directs au système ou recevoir des notifications sur la défaillance du système.

## 3.2. Description des différents cas d'utilisation

A partir des rôles déterminés ci-dessus, nous allons maintenant définir le comportement souhaité du logiciel, en définissant ses différents cas d'utilisation.

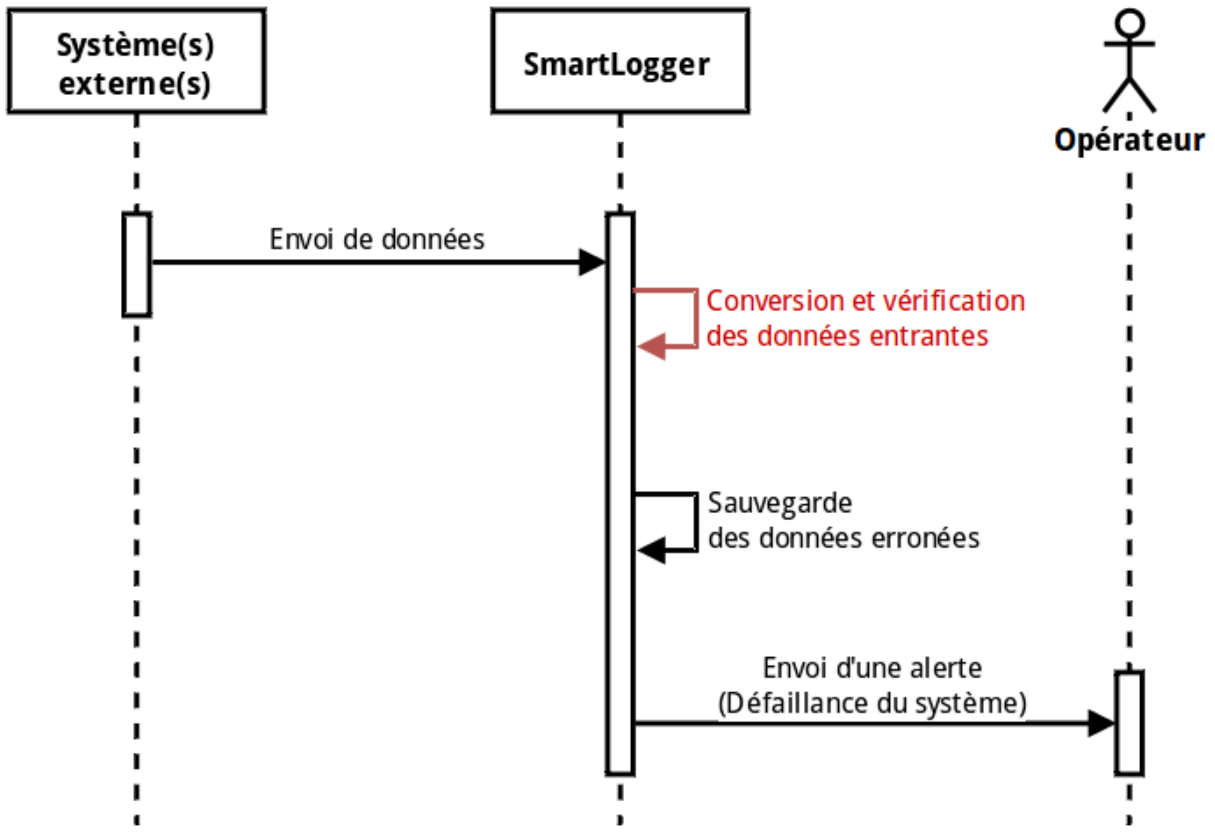


## UC-001 : Fonctionnement principal



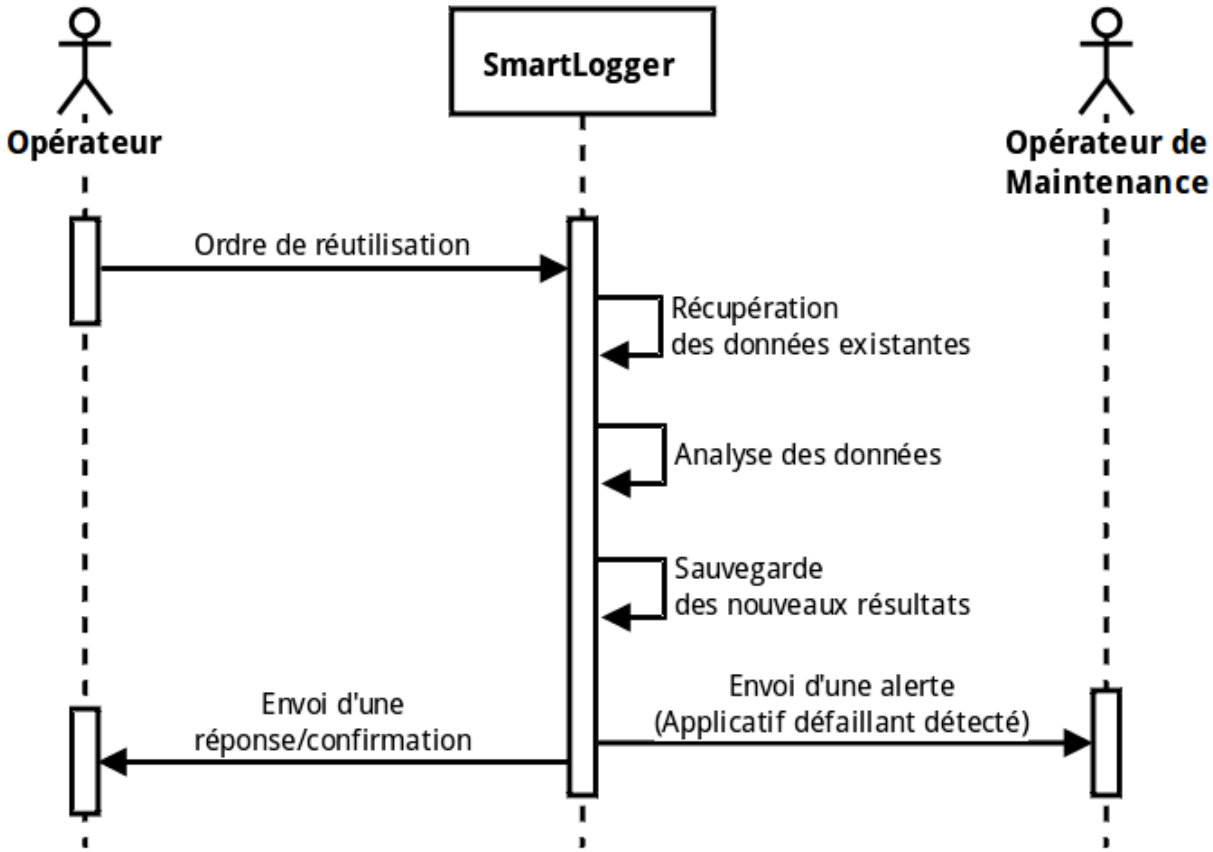

| | |
|---|---|
| Acteurs concernés | Le système, un/des système(s) externe(s), et des opérateurs de maintenance sur ce(s) système(s). |
| Description | Cas d'utilisation de la machine lors de son mode de fonctionnement principal.<br><br>La machine reçoit des données d'un système externe qui sont converties, vérifiées, puis stockées dans la base de données.<br>Elle effectuera alors son analyse et utilisera les résultats pour affiner son comportement.<br><br>Si une alerte critique est détectée, elle procédera à l'alerte des opérateurs associé à l'applicatif défaillant. |
| Pré-conditions | a) Le système est opérationnel.<br>b) Le système dispose de données à traiter.<br>c) Le système est inactif (ne réalise aucune autre tâche). |
| Événement déclenchant | Réception d'un volume de données quelconque provenant d'un système externe. |
| Conditions d'arrêt | a) Le système termine son cycle d'exécution.<br>b) Une alerte a été lancée, suite au traitement des données. |

## UC-002 : Détection de données erronées

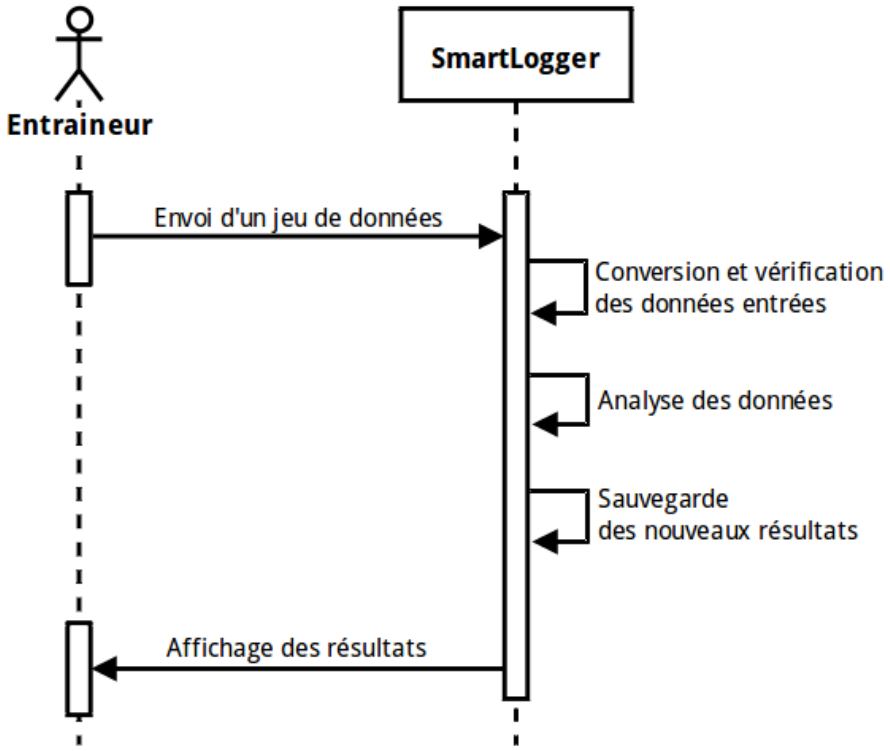



| | |
|---|---|
| Acteurs concernés | Le système, un de ses opérateurs et un ou des système(s) client(s). |
| Description | Intervient lorsqu'un système externe envoie des données erronées ou incohérentes au système.<br><br>Si une erreur apparaît lors de la conversion ou de la vérification des données, on procède à leur stockage en attendant leur possible exploitation future (format inconnu, etc.)<br><br>On envoie également une alerte afin de notifier les opérateurs de cette anomalie. |
| Pré-conditions | a) Le système est opérationnel.<br>b) Le système dispose de données à traiter.<br>c) Le système est inactif (ne réalise aucune autre tâche). |
| Événement déclenchant | a) La conversion des données a échoué.<br>b) La vérification de la validité des données a échoué. |
| Conditions d'arrêt | a) Les données ont été stockées dans la base de données.<br>b) Une notification à un/des opérateur(s) a été envoyée. |

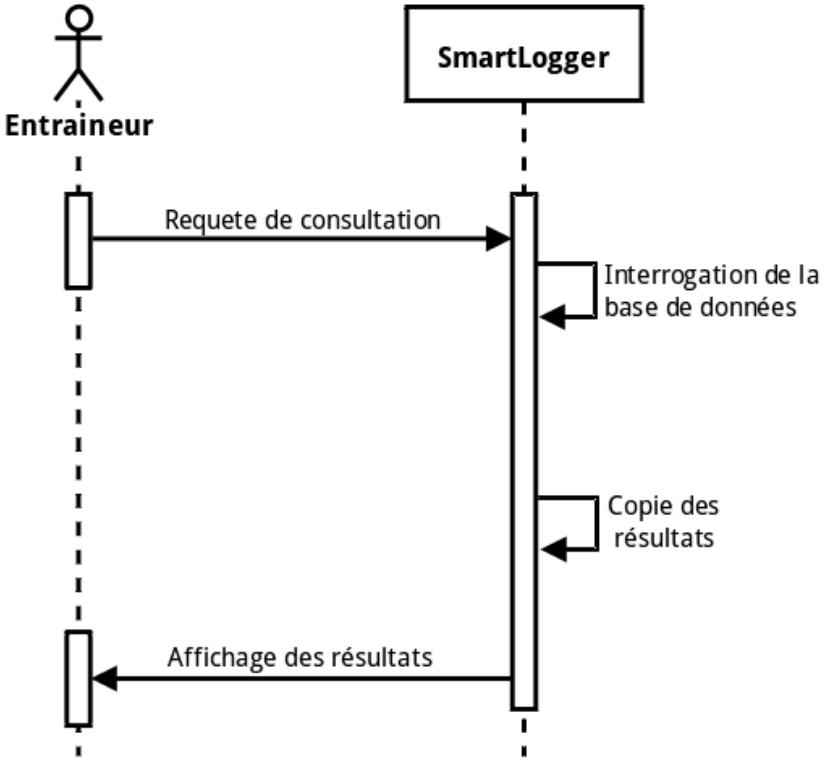
## UC-003 : Réutilisation de données stockées




| | |
|---|---|
| Acteurs concernés | Un opérateur, le système |
| Description | Cas d'utilisation correspondant à un ordre direct d'un opérateur.<br>Intervient lorsqu'on souhaite effectuer des traitements sur des données auparavant incorrectes ou pour réévaluer le comportement du système.<br><br>Le système va effectuer un traitement sur l'échantillon de données désigné par l'opérateur. Le fonctionnement est alors identique à 4.1. |
| Pré-conditions | a) Le système est opérationnel.<br>b) Le système dispose de données à traiter.<br>c) Le système est inactif (ne réalise aucune autre tâche). |
| Événement déclenchant | a) Ordre d'envoi de  données invalides.<br>b) La conversion des données a échoué.<br>c) La vérification de la validité des données a échoué. |
| Conditions d'arrêt | a) Les données sont stockées dans la base de données.<br>b) Une alerte est envoyée. |

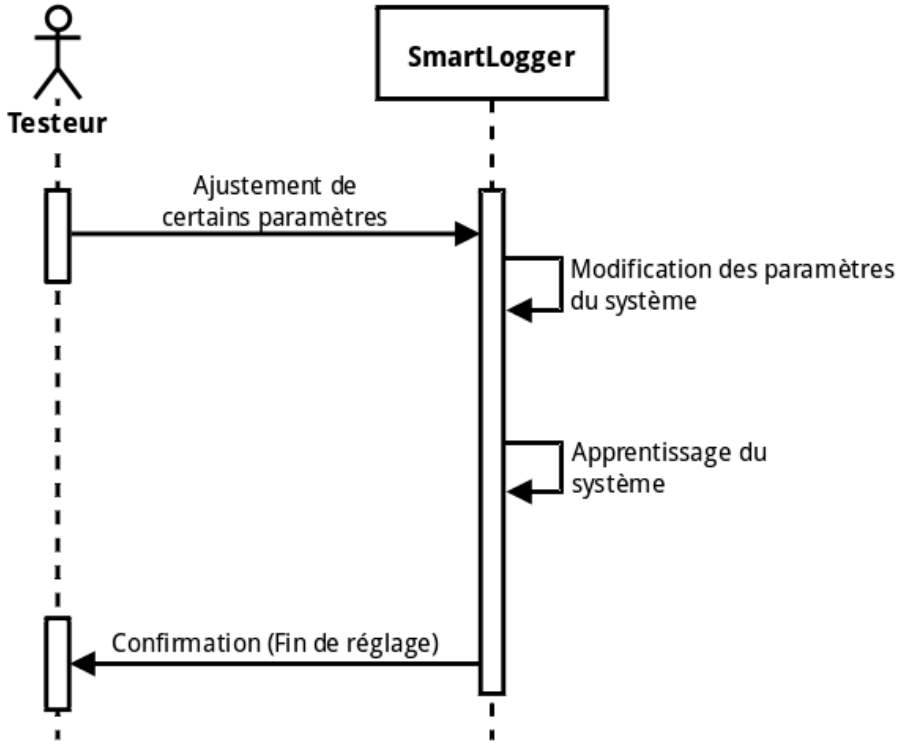
| UC-004 : Entraînement du système | |
|---|---|
|  | |
| Acteurs concernés | Un entraîneur, le système |
| Description | Mode de fonctionnement correspondant à l'entraînement du système (perfectionnement de son module d'analyse).<br><br>A cet effet, l'opérateur entre un jeu de données au système. Ce dernier effectue alors le même processus qu'en 4.1, à la différence qu'il affiche les résultats sur l'interface graphique. |
| Pré-conditions | a) Le système est opérationnel.<br>b) La base de données possède des données affichables.<br>c) Le système est inactif (ne réalise aucune autre tâche). |
| Événement déclenchant | Action de l'opérateur sur l'interface. |
| Conditions d'arrêt | Fin des processus d'affichage et d'apprentissage. |

## UC-005 : Consultation des données




| | |
|---|---|
| Acteurs concernés | Un entraîneur, le système |
| Description | Dans ce cas de figure, l'entraîneur peut consulter la base de données afin de vérifier la cohérence de traitements précédemment réalisés par le système. |
| Pré-conditions | a) Le système est opérationnel.<br>b) La base de données possède des données affichables.<br>c) Le système est inactif (ne réalise aucune autre tâche). |
| Événement déclenchant | Action de l'opérateur sur l'interface. |
| Conditions d'arrêt | Fin du processus d'affichage. |

## UC-006 : Ajustement du comportement




| | |
|---|---|
| Acteurs concernés | Un testeur, le système. |
| Description | Correspond à l'ajustement de l'analyse de la machine.<br><br>L'interface sert alors d'intermédiaire entre l'opérateur et le système, lui permettant de réaliser des modifications.<br>Ce dernier les applique puis, confirme leur application. |
| Pré-conditions | a) Le système est opérationnel.<br>b) Le système est inactif (ne réalise aucune autre tâche). |
| Événement déclenchant | Action de l'opérateur sur l'interface. |
| Conditions d'arrêt | La machine a confirmé les différentes modifications effectuées. |

## 4. Exigences opérationnelles

| P | Référence | Désignation |
|---|---|---|
| | OP-1 | **Le système doit fonctionner sans interruption** |
| | OP-1.1 | Les opérations de modification du système (maintenance, extension...) doivent être simples et rapides à mettre en place. |
| | OP-1.2 | Le nombre d'arrêts du système doit être le plus faible possible, sur toute la période de sa durée de vie. (Hors opérations d'ajout de modules externes) |
| | OP-1.3 | Le système doit être conçu pour fonctionner sur un serveur de type Linux. |
| | OP-2 | **Le traitement des résultats doit être adapté à la criticité détectée** |
| | OP-2.1 | Les détections de failles critiques doivent mener à une levée d'alerte |
| | OP-2.2 | Les détections de failles mineures doivent être signalées avec une importance moindre qu'une alerte. |
| | OP-2.3 | Les résultats sans importance ne doivent pas être signalés |
| | OP-3 | **Toute donnée manipulée par le système doit être enregistrée** |
| | OP-3.1 | Les données incohérentes ou non vérifiées seront stockées sans traitement préalable |
| | OP-3.2 | Les résultats issus d'une analyse seront stockés avec les données brutes correspondantes |
| | OP-3.3 | Un utilisateur du système doit pouvoir manipuler d'anciennes données déjà utilisées |
| | OP-4 | **Le système doit fonctionner de manière cyclique** |
| | OP-4.1 | Les résultats des analyses seront envoyés à intervalle régulier |
| | + OP-4.1.1 | Un script devra assurer l'envoi de résultats toutes les X secondes |
| | + OP-4.1.2 | Même si une analyse est en cours, le système doit procéder à un envoi de l'ensemble des résultats obtenus |
| | OP-4.2 | Chaque cycle d'exécution doit commencer par un traitement des données reçues |
| | OP-4.3 | Chaque cycle d'exécution doit se terminer par un apprentissage des données traitées |
| | OP-5 | **Le système doit prévenir toute tentative d'utilisation incorrecte des données** |
| | OP-5.1 | La consultation des données du système ne peut s'effectuer qu'au travers de l'interface |
| | OP-5.2 | Les données traitées doivent subir un chiffrement avant leur stockage dans la base de données |
| | OP-5.3 | Le système doit vérifier l'intégrité des données lues en entrée, avant de procéder à toute autre phase de traitement |

Échelle de priorité : Indispensable, Important, Optionnel

## 5. Exigences d'interface

| P | Référence | Désignation |
|---|---|---|
| | IN-1 | **L'interface d'entrée du système doit pouvoir traiter tout type de log** |
| | IN-1.1 | L'interface d'entrée doit offrir la possibilité d'utiliser plusieurs types de données prédéfinis |
| | + IN-1.1.1 | Le système doit pouvoir utiliser des données de type Apache Kafka |
| | + IN-1.1.2 | Le système doit savoir manipuler des données de type Shinken |
| | + IN-1.1.3 | Le système doit pouvoir utiliser des données logstash |
| | IN-1.2 | L'interface d'entrée doit pouvoir être modifiée afin d'accepter de nouveaux types de données |
| | IN-2 | **L'interface de sortie doit disposer de plusieurs fonctionnalités d'alerte et de notification** |
| | IN-2.1 | L'interface de sortie doit disposer de plusieurs fonctionnalités prédéfinies afin d'effectuer ses alertes |
| | + IN-2.1.1 | L'alerte doit pouvoir être donnée via l'application Slack |
| | + IN-2.1.2 | L'alerte doit pouvoir être donnée par Mail |
| | + IN-2.1.3 | L'alerte peut être donnée par SMS |
| | IN-2.2 | L'interface de sortie doit pouvoir s'adapter à des formats de sorties supplémentaires |
| | IN-3 | **Le système doit proposer une interface afin d'y consulter ses données** |
| | IN-3.1 | L'interface doit permettre de consulter conjointement un log et son résultat d'analyse (produit par le système). |
| | IN-3.2 | L'interface doit permettre de classer les données consultables en fonction de critères variés. |

Échelle de priorité : Indispensable, Important, Optionnel

## 6. Exigences de qualité logicielle et de réalisation

| P | Référence | Désignation |
|---|---|---|
| | QR-1 | **La compréhension du code source doit être facilitée** |
| | QR-1.1 | Le code source fourni doit être propre et commenté |
| | QR-1.2 | Les identificateurs de variables et de fonctions doivent être significatifs de leur utilité |
| | QR-1.3 | Les identificateurs utilisés doivent être issus de la langue anglaise |
| | QR-1.4 | Le code source final devra être correctement documenté |
| | QR-2 | **Le type de programmation employé doit être le plus modulaire possible** |
| | QR-2.1 | Les nouveaux types de flux d'entrées doivent être facilement implémentables, au moyen de nouveaux modules d'entrée |
| | QR-2.2 | L'ajout de nouvelles méthodes d'alerte ou de notification souhaitées, doit pouvoir s'effectuer de façon simple, en ajoutant de nouveaux modules de sortie. |
| | QR-2.3 | Les modules algorithmiques gérant la partie d'analyse prédictive du système doivent être interchangeables. |

Échelle de priorité : Indispensable, Important, Optionnel