**Mission :**

La R&D a pour responsabilité de fournir l'ensemble des moyens de simulation numérique nécessaires à la gestion des portefeuilles de produits financiers de MARK. A cet effet, elle met en œuvre toutes les technologies dont elle peut disposer : compilation de code à la volée, calcul vectoriel et sur GPU, grilles de calcul internes ou dans le cloud public, etc. La R&D a développé une plateforme de calcul distribué à destination de plusieurs sites internes et externes de distribution de produits financiers dérivés. Ce service, basé sur une architecture micro-service et déployé dans le cloud public, reçoit plusieurs milliers de requêtes de calcul par jour. Certaines de ces requêtes ne peuvent aboutir, du fait d'une erreur technique ou parce qu'elles étaient malformées. Chaque fois, un message d'alerte est émis, qui nécessitera un traitement par l'équipe de production ou l'équipe de développement, en fonction du type d'erreur.

**Reformulation de la mission :**

Les membres de l’équipe de Bender reçoivent des milliers d’erreurs/alertes par jour, ce qui rend le suivi et la détection des patterns des alertes quasi-impossible, et des fois des nouvelles alertes passent sous le radar, sans être vues par les membres de l’équipe.

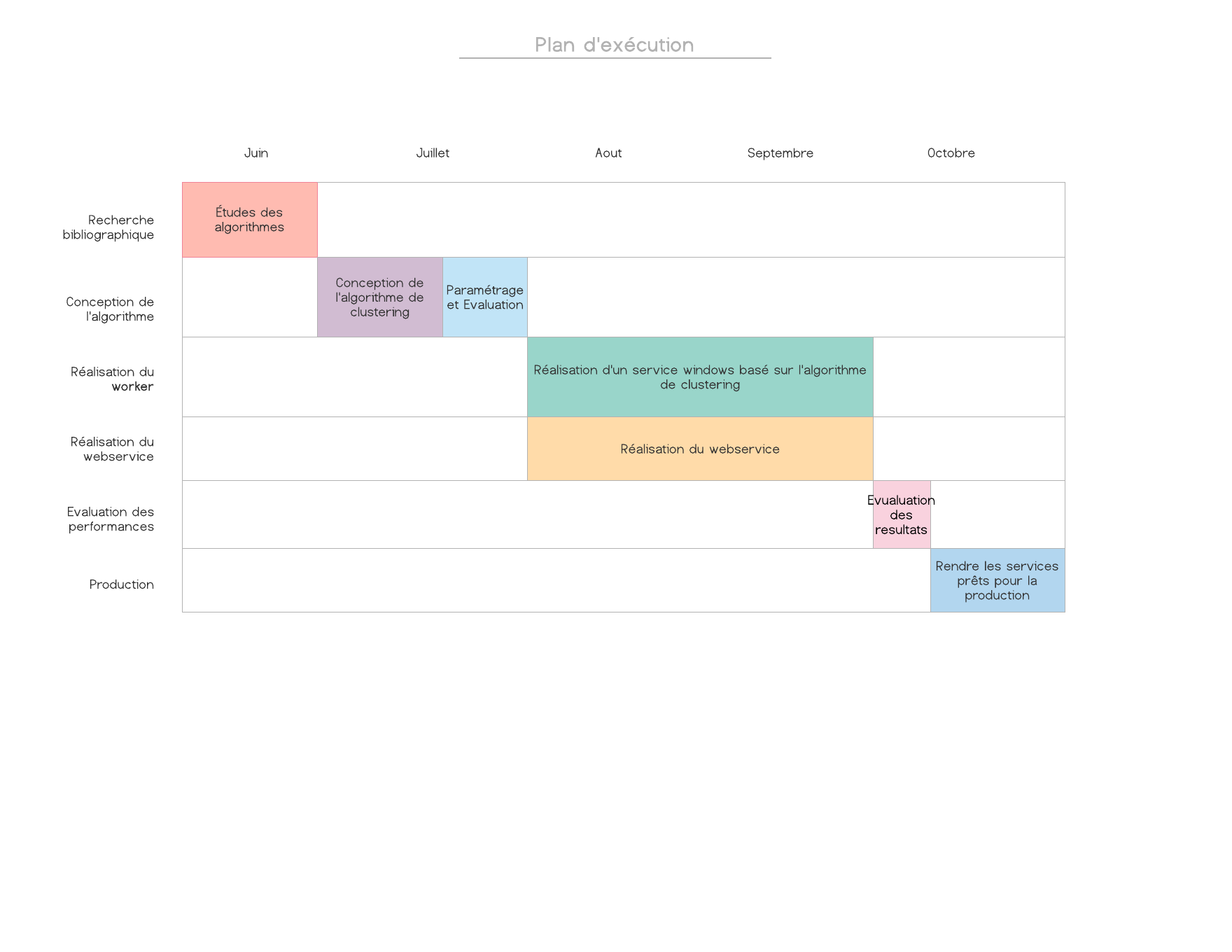
Les alertes ont des formes différentes, deux alertes qui ont la même cause peuvent avoir deux messages d’erreur différents, et partant une comparaison simple des messages ne reprendra pas au besoin.

Les alertes ne sont pas définies, par conséquent, on peut traiter des nouvelles alertes, qui n’étaient jamais traites lors de l’entrainement du model, durant la production. Par la suite, il faut que l’algorithme crée une nouvelle classification pour ces alertes, ce qui signifie que notre algorithme ne devrait pas se baser sur le nombre de cluster comme paramètre (exemple K-means qui prend le nombre de classifications comme paramètre), et que l’algorithme soit incrémental (qui peut s’entrainer au fur et au mesure de l’exécution)

**Objectif :**

Réalisation d’un outil de classification, suivi automatique et visualisation de l’évolution de ces alertes (le model de classification, les services permettant l’intégration du model dans l’architecture micro-service de la société général).

**Plan d’exécution :**



**La mission est repartie en 6 grandes phases :**

**Recherche bibliographique** | **FAIT** : Étude des algorithmes de classification existants et les solutions qui traitent les logs (journal de messages).

**Conception de l’algorithme de classification** | **FAIT** : Après avoir fait le choix de l’algorithme de classification, **DBSCAN**, il fallait faire des changements pour le rendre incrémental (Spécifications) et de concevoir les fonctions qui assure le bon fonctionnement du model a titre d’exemple : fonction de calcul de distance entre les alertes, la classification incrémentale, la gestion des clusters …

Diagram

Description automatically generated

Figure : Architecture de la solution **Benster**

**Réalisation du workeur** | **FAIT** : Le workeur est un service Windows qui utilise le model/algorithme de classification et qui consulte le service distant pour récupérer les nouvelles alertes et les classifier avec une fréquence donnée. Ainsi, lors de cette partie nous avons réalisé toutes les fonctions qui permet le service de communiquer avec le système d’exploitation Windows comme étant un service Windows. Nous avons conçu le modèle de communication avec le service distant, les trainements à faire après la récupération des alertes, les conversions de données, et la communication avec la base de données.

**Réalisation du Web service** | **FAIT** : le WebService est le service qui permet la communication avec la base de données et qui fournit les fonctionnalités attendues (voir le diagramme de cas d’utilisation de BENSTER).

**Évaluation des résultats** | **En cours** : le lancement d’une classification en commençant par 01/01/2020, la première partie des données du 01/01/2020 jusqu’au 31/08/2020 est stockée localement, la deuxième partie des données à partir du 01/09 est récupérée depuis le service distant lors la classification. Lors de cette partie il faut identifier les erreurs de classification, les causes de la latence du traitement, la vitesse avec laquelle faut appeler le service distant sans causer une panne.

**Production** | **Pas encore** : Lors de la dernière étape, il faut rendre les services prêts pour la production en ajoutant des tests unitaires et des tests d’intégration, et en utilisant des variables d’environnement au lieu des fichiers de configuration stockés localement.

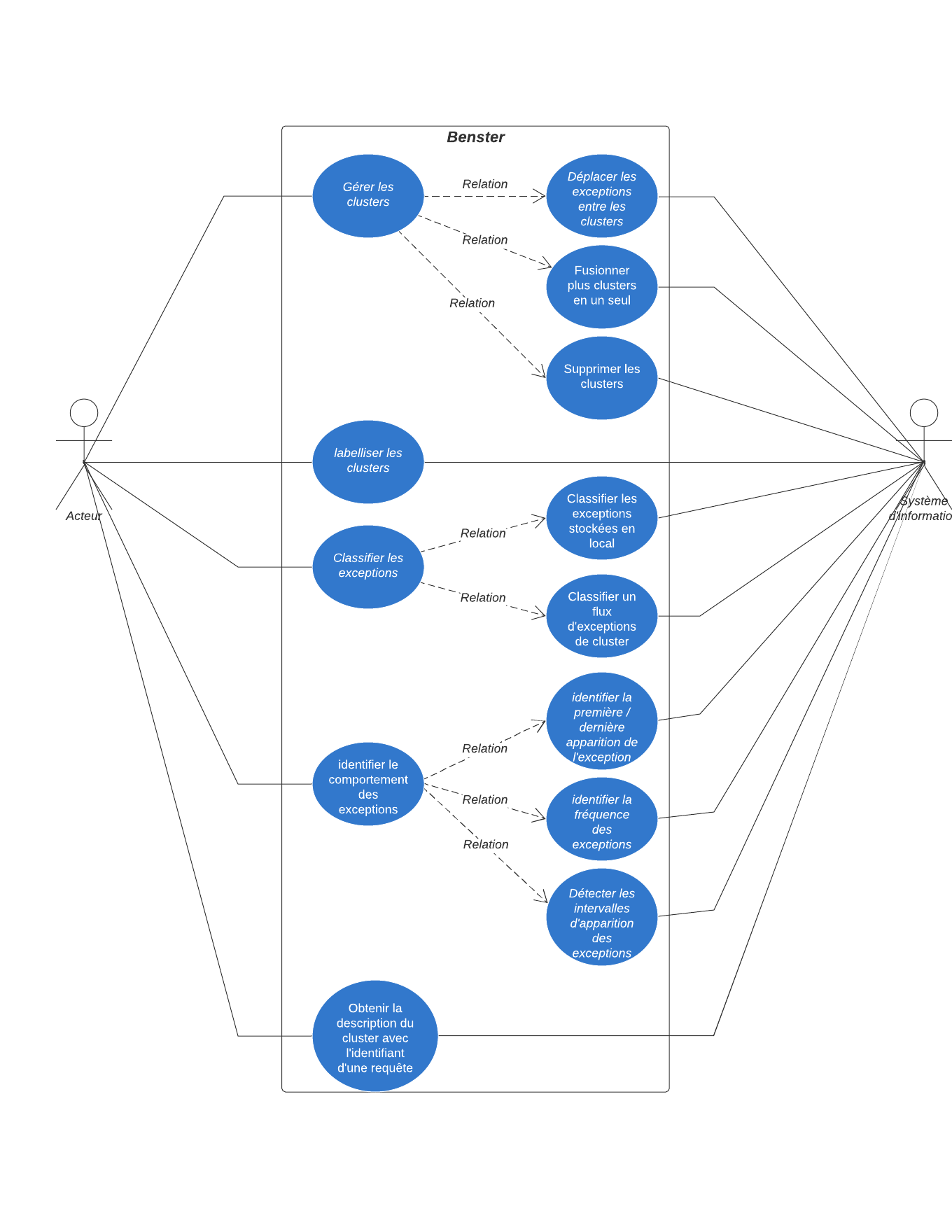


Figure : diagramme de cas d’utilisation de la solution **BENSTER**