Institut Paul Lambin

BAC 2 Informatique de gestion

Cahier des charges

Outil de gestion des bugs et des demandes d'améliorations

Auteurs :
Christopher Sacré
Antoine Maniet
Alexandre Maniet
Timothé Bouvin
Damien Meur

Professeur :
B. Lehmann

6 décembre 2016

Table des matières

1 Contexte

La société *SoftDevelop S.A.* développe trois lignes de produits-logiciels (*Soft Clients*, *Soft Fournisseurs* et *Soft Compta*) vendus par des intermédiaires. Chaque intermédiaire a un contact commercial chez SoftDevelop.

C'est notamment via cet intermédiaire commercial que le client prend contact avec l'entreprise elle-même. La prise de contact peut avoir différentes raisons : demande d'amélioration des produits, correction de bugs ou encore demande d'information. C'est le commercial qui fait ensuite parvenir les demandes aux équipes de développement.

De ce fait, la société *SoftDevelop S.A.*souhaite sous-traiter la création d'un nouveau programme leur permettant d'automatiser la gestion des demandes d'améliorations et des bugs. Ce cahier des charges a pour but de présenter la solution proposée par notre entreprise.

2 Analyse de l'existant

2.1 Acteurs

Lors de l'analyse du système actuel de *SoftDevelop S.A.*, 3 acteurs se sont démarqués pour la gestion des bugs.

Le commercial : il fait le lien entre l'équipe de développement et le client. C'est via le commercial que le client fait savoir qu'un bug est apparu.

Le membre de l'équipe : il s'agit de la personne qui s'occupera de corriger le bug, de réaliser des tests.

Le responsable : il supervise plusieurs membres d'équipes et s'occupe de vérifier si le travail est correctement effectué et si le bug est corrigé.

Une autre dénomination revient fréquemment : **l'utilisateur** , cette dernière est utilisée désigner les 3 acteurs cités précédemment en même temps et sans faire de distinction au niveau des rôles.

2.2 Documents manipulés

Actuellement, la transmission des bugs se fait via e-mail et à l'aide d'une fiche signalétique et d'une fiche descriptive.

2.2.1 Fiches descriptive et signalétique

Exemple fiche descriptive	Exemple fiche signalétique
* Hardware : Nom de la station: Description PC : Version OS: * Utilisateur a-t-il les droits d'administrateur sur le PC : OUI - NON * Paramètres régionaux : * Que s'est-il passé : ** Comportement : ** Message d'erreur : * Reproductibilité : * Toujours / parfois / quelques fois / expérimenté une seule fois * Commentaires: * Etapes détaillées pour reproduire le bug : * Remarques éventuelles :	*Logiciel : **Type de logiciel : Soft Clients / Soft Fournisseur / Soft Compta **Version du logiciel : **Version corrigée : *Nom intermédiaire : *Priorité : *P1 / P2 / P3 *Commentaires : *Sévérité : *Critique / Majeure / Modérée / Mineure *Commentaires : *Résumé :

TABLE 1 – Exemple de fiches descriptive et signalétique

Explication des différentes priorités :
- **P1 :**la priorité la plus haute, réservée pour les demandes urgentes.
- **P2 :** la priorité moyenne.
- **P3 :** la priorité la plus basse.

Explication des différentes sévérités :
- **Critique :** la sévérité la plus haute ; elle indique que le bug est bloquant et empêche le fonctionnement du logiciel et donc la production chez les clients.
- **Majeure :** elle indique que le bug est grave mais qu'il existe une façon de travailler pour le contourner.
- **Modérée :** elle indique que le bug n'empêche pas le logiciel de fonctionner ni le client de travailler.
- **Mineure :** il s'agit d'un bug « cosmétique » qui sera résolu en dernier lieu.

2.2.2 E-mail

Par ailleurs, tous les commentaires se font actuellement via e-mail. C'est d'ailleurs via ce même e-mail que l'on suivra l'évolution du bug et que l'on

gardera une trace de ce de dernier.

2.3 Règles de gestion

2.3.1 Détection du bug

Le bug est détecté / l'amélioration est demandée le plus souvent par les utilisateurs qui font parvenir cette information au commercial avec qui il est en contact. C'est le commercial qui s'occupe d'encoder le bug / l'amélioration dans le système et c'est donc ce dernier qui lance le processus de gestion.

2.3.2 Gestion des commentaires

Actuellement les commentaires sont gérés par e-mail. Tous les commentaires concernant un bug se rattachant au mail signalant le bug en question. De plus, l'évolution du projet se fait également via e-mail.

2.3.3 Modification des documents

À tout moment un membre d'équipe ou un responsable peut modifier *la fiche signalétique du bug* :

- le logiciel sur lequel il a rencontré le bug
- le nom de l'intermédiaire
- la version du logiciel dans laquelle le bug est survenu
- la version du logiciel dans laquelle le bug a été corrigé
- la priorité
- la sévérité
- le résumé

Il faudra donc permettre la modification de ces informations par le responsable et ses membres d'équipes.

Par opposition, *la fiche descriptive d'un bug* ne pourra jamais être modifiée. Dans le cas où cette dernière serait erronée ou contiendrait une descrpition peu fiable ou devrait être tout simplement modifiée, cela se fera à l'aide de commentaires.

2.3.4 Subdivision en différents états

La gestion d'un bug [1] se fait à l'aide de différents états, un bug ayant toujours un état à un moment donné :

- Nouveau
- Confirmé
- Pris en charge
- Corrigé
- Vérifié

1. Nous utiliserons le mot ≪ bug ≫pour se référer à un bug ou à un demande d'amélioration RFE dans le reste de ce cahier des chages.

— Clos
— Rejeté
— Non-reproductible
— Doublon
— Rouvert

Le passage d'un état à l'autre symbolise un changement dans la façon de traiter ce bug et son avancée dans le processus de résolution. Il faudra donc notifier les utilisateurs de l'évolution du projet via ces différents états. [2]

2.3.5 Finalisation du traitement

Seul le responsable peut décider si un bug est corrigé ou non. C'est ce dernier qui prendra la décision finale et qui mettra fin au traitement en livrant la version corrigée au client. Par ailleurs un bug pourra très bien être rouvert par la suite.

3 Objet du marché

La société *SoftDevelop S.A.* souhaite sous-traiter le développement d'un outil de gestion des bugs et des demandes d'améliorations, pour ses logiciels.

3.0.1 Objectifs

— Signaler un bug ou RFE
— Faire évoluer le bug ou le RFE tout au long de son cycle de vie
— Rechercher un bug ou un RFE sur base de son identifiant unique ou de toute information de sa fiche signalétique
— Visualiser l'ensemble des bugs ou RFE correspondant aux critères de recherche, sous un format de tableau, chaque ligne correspondant à un bug
— Livrer un mode d'emploi avec l'outil de gestion des bugs

3.0.2 Contraintes

— La maintenance de l'outil de gestion des bugs et des demandes d'améliorations sera assurée pendant trois ans
— L'application ne sera développée qu'en utilisant des standards ouverts du marché tant au niveau de la programmation que du langage de programmation

2. Si un bug clos veut être rouvert, on ne changera pas l'état de ce dernier, mais on considèrera qu'il s'agit d'un nouveau bug.

4 Spécifications applicatives

4.1 L'application SoftRepair™

TO DO = INTRODUCTION

4.2 Description des fonctionnalités

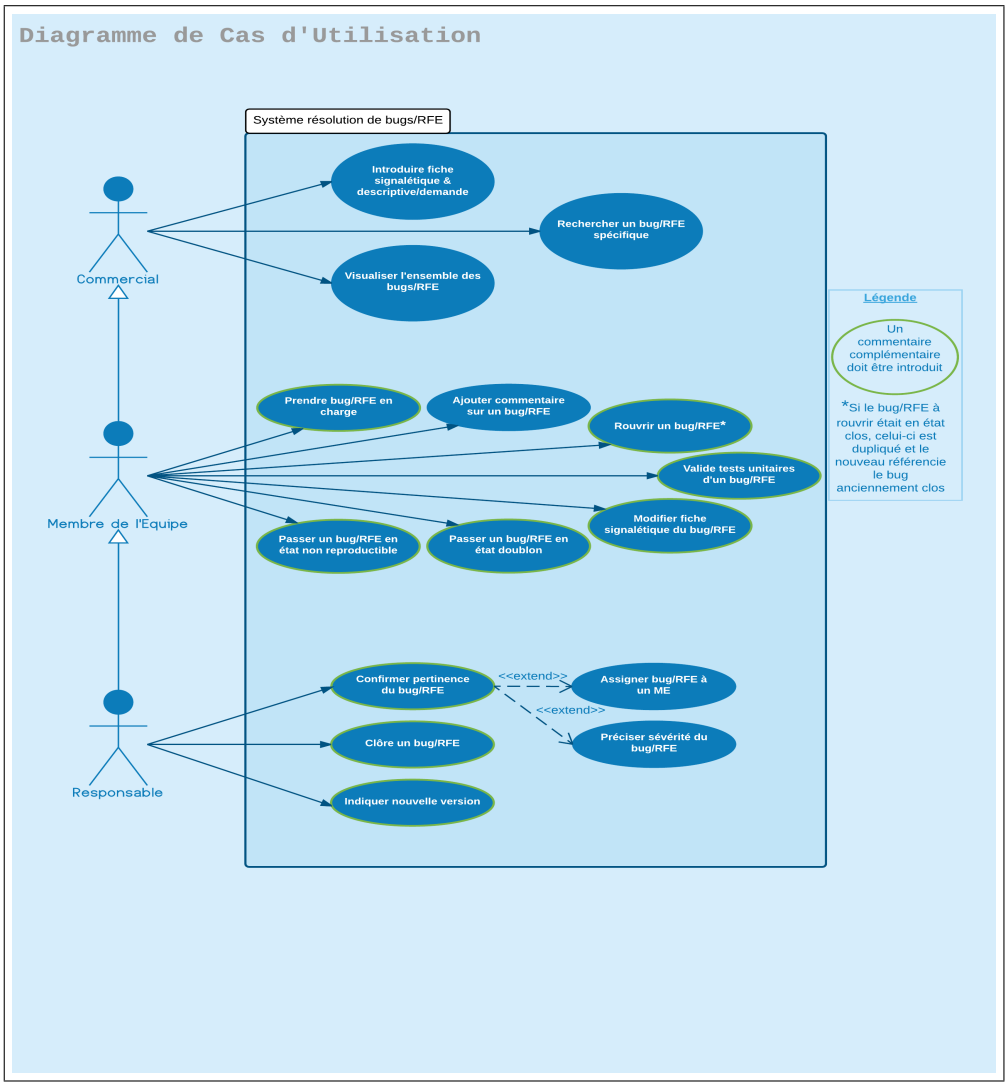

FIGURE 1 – Diagramme des cas d'utilisation

4.3 Flux d'information

4.3.1 Diagramme de flux d'informations

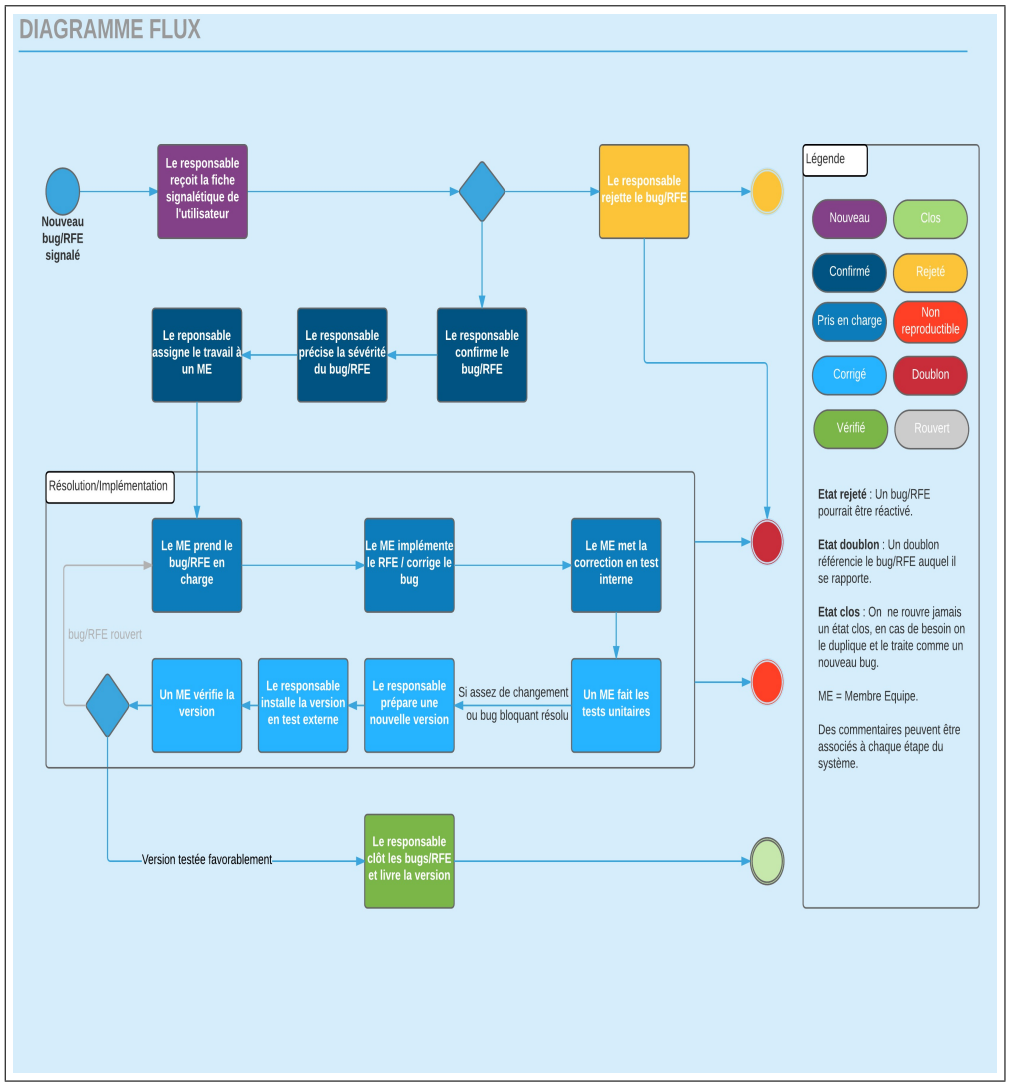

FIGURE 2 – Diagramme de flux de la gestion d'un bug

4.3.2 Description du flux d'information

Lorsqu'un nouveau bug est constaté par un intermédiaire (l'un des clients de SoftDevelop S.A.) celui-ci le fais parvenir via le commercial en remplissant

une fiche signalétique ainsi qu'une fiche descriptive décrivant différents points du bug. Le responsable va par la suite vérifier sa pertinence (il répondra en outre à la question : *"l'équipe va t'elle traiter ce bug / RFE"*).

Nous distinguerons à la suite de cette étape deux cas distincts : le responsable rejette le bug / RFE et le responsable confirme le bug / RFE. Dans le cas où le bug est rejeté, ce dernier restera dans le système et pourra par la suite être réactivé. Au contraire si celui-ci est confirmé alors le responsable précisera la sévérité du bug / RFE et l'assignera à un membre de son équipe. Le membre d'équipe assigné au bug va par la suite prendre le bug en charge (on entend par là que le membre d'équipe va corriger le bug et implémenter sa solution au sein du sytème).

Dès que le bug est corrigé, le membre d'équipe devra l'indiquer au reste de son équipe et documenter sa correction. Ensuite lorsqu'une ou plusieurs corrections de bugs doivent être livrées (Si il y a eu assez de changements pour faire une nouvelle version ou si un bug bloquant à été résolu), le responsable des développements prépare une nouvelle version et installe cette nouvelle version dans l'environnement de tests externes. Par la suite, un membre d'équipe pourra choisir un bug installé dans l'environnement de tests externes et le tester. Si le bug est testé favorablement, le membre d'équipe indique que celui-ci est vérifié et décrit les différents tests qu'il a effectué avant de parvenir à sa décision. Si au contraire un bug n'est pas

4.4 Description du travail des utilisateurs

4.4.1 L'intermédiaire

L'intermédiaire n'est pas un utilisateur actif de l'application, sa fonction est de faire le relais entre les clients et les bugs qu'ils rencontrent et le commercial qui les signale à l'équipe de développement. Pour cela, l'application prévoit une utilsiation de suivi de bug pour que l'intermédiaire puisse informer un client de l'avancement de la résolution d'un bug en particulier. L'intermédiaire **n'est pas authentifié** au sein de l'application. N'importe quel utilisateur ayant accès à un bug concernant un intermédiaire peut en seul geste lui envoyer un e-mail généré automatiquement résumant l'état du bug actuel.

NOTE : AJOUTER UNE IMAGE DU BOUTON FAIRE PART A L INTERMEDIAIRE + UN EXEMPLE DE MAIL GENERE AUTOMATIQUEMENT

4.4.2 Le commercial

Le commercial est un utilisateur **authentifié** de l'application (grâce à son identifiant unique et son mot de passe). Il est le point de relais entre l'intermédiaire et l'équipe de développement, c'est lui qui signale tous nouveaux bugs.

Signalement de bug Pour cela il suit suffit depuis l'écran principal d'appuyer sur le bouton « Ajouter un bug /RFE », l'application lui invitera ensuite à remplir un formulaire correspondant à la fiche descriptive du bug.

Note : placer ici la vue avec la fiche descriptive et signalétique d'un bug (en mettant en évidence le boutton ajouter bug) et en grisant la possibilité de changer la sévérité d'un bug

Visualisation et recherche de bugs Pour l'aider dans sa tâche, l'application lui permettra de visualiser l'ensemble des bugs et d'effectuer des recherches (par état, par logiciel , le concernant ou pas... A VERIFIER/ PRECISER COHERENCE GRAPHIQUE IHM).

Note : placer ici la vue principal en insistant sur la partie visualistion/recherche

Espace personnel Le commercial à également accès à son propre espace personnel depuis lequel il peut modifier ses identifiants de connexion à l'application. Depuis cet écran, il peut avoir accès à un certain nombre de statistiques le concernant. (A VERIFIER/ PRECISER COHERENCE GRAPHIQUE IHM)

4.4.3 Le membre d'équipe de développement

Le membre d'équipe tient un rôle centrale dans l'application, c'est lui qui gèrera la majorité des transitions d'état des bugs afin de pouvoir garder un détail sur l'évolution de ceux-ci. Il dispose également des mêmes fonctionnalités que le commercial. La fonction de recherches de bugs possède une option supplémentaire par rapport au commercial qui lui permet de voir rapidement quels bugs lui ont été assignés par son responsable. A VERIFIER/ PRECISER COHERENCE GRAPHIQUE IHM)

Insertion de commentaires Pour l'aider dans le développement du bug et pour tenir informer le reste de l'équipe, le membre d'équipe peut laisser un commentaire grâce à une vue prévue à cet effet. Cette action est obligatoire lorsqu'il change l'état d'un bug (un champ est directement prévu dans le menu de transition d'état pour les commentaires obligatoire ainsi que dans la fenêtre commentaires pour les facultatifs).Il est à noter que chaque commmentaire lié à un bug conserve l'état du bug durant lequel il a été écrit. La vue ci-dessous donne un exemple de l'historique des commentaires d'un bug. Un code couleur correspondant à l'état des bugs indique durant quel étape le commentaire a été écrit. Note : placer ici la vue concernant les commentaires

Transition d'état Le menu principal du bug comprend différents boutons pour marquer la transition d'état d'un bug.

Le membre d'équipe peut ainsi choisir de marquer un bug comme

— *pris en charge* : il décide de prendre en charge son développement.

— *corrigé* : il considère que le bug est corrigé après avoir lui-même effectué ses tests unitaires.
— *validé* :il a terminé de valider les tests sur le bug en question dans l'environnement de tests de la nouvelle version (comprenant la version corrigée du bug.
— *non-reproductible* :le bug est considéré comme annulé.
— *rouvert* : si un bug non-reproductible est remis en route.
— *doublon* : si le bug a déjà été signalé.
— *clos* : si une version reprenant le bug a déjà été livré.
— *nouveau* : si une version reprenant le bug a déjà été livré.
— assigné ?

. Note : placer ici la vue concernant les transitions de bug (griser celui de clore pour responsable) + monter qu'on peut bel et bien référence le bug initial pour létat doublon

Modification d'une fiche Un membre d'équipe peut modifier la fiche signalétique ou descriptive d'un bug à tout moment. Note : placer ici la vue où l'on peut voir un bouton pour la modification d'une fiche

4.4.4 Le responsable d'équipe de développement

Le responsable d'équipe qui est également un utilisateur **authentifié** est doté des mêmes fonctions que le membre d'équipe. A cela près qu'il est doté d'outils supplémentaires liés aux préparation de nouvelles versions (pouvant regrouper plusieurs bugs) ainsi qu'à l'assignation de bugs à des membres d'équipes.

Transitions d'états Trois transitions d'états lui sont réservées « confirmé »et « assigné »et « clôs ». En effet le responsable peut dans un premier temps confirmer un nouveau bug en précisant sa sévérité mais décider de l'assigner à un membre d'équipe à un autre moment. L'état « clôs »correspond à l'état d'un bug lorsque le responsable confirme la correction de celui-ci et qu'il s'apprête à préparer une nouvelle version du logiciel prenant en compte les modifications effectuéesA VERIFIER/ PRECISER COHERENCE GRAPHIQUE IHM +DSD + DIAGRAMME UC + DIAGRAMME FLUX) Note : placer ici la vue où l'on peut voir le champs sévérité de la fiche signalétique non grisé + la possibilité de sélectionner un membre d'équipe pour lui assigner un bug (attention différent prise en charge, il faut rajouter un nouvel état ==¿ à discuter)

Préparation de nouvelles versions Une fois qu'un ou plusieurs bugs dont l'état est « clôs »le responsable peut décider de préparer une nouvelle version du logiciel concerné. Pour l'aider dans sa tâche, un système de glisser-déposer lui permettra de reprendre les bugs repris dans la nouvelle version aisément. Note : placer ici la vue où l'on prépare une nouvelle version, avec le glisser-déposer

4.5 Description des données

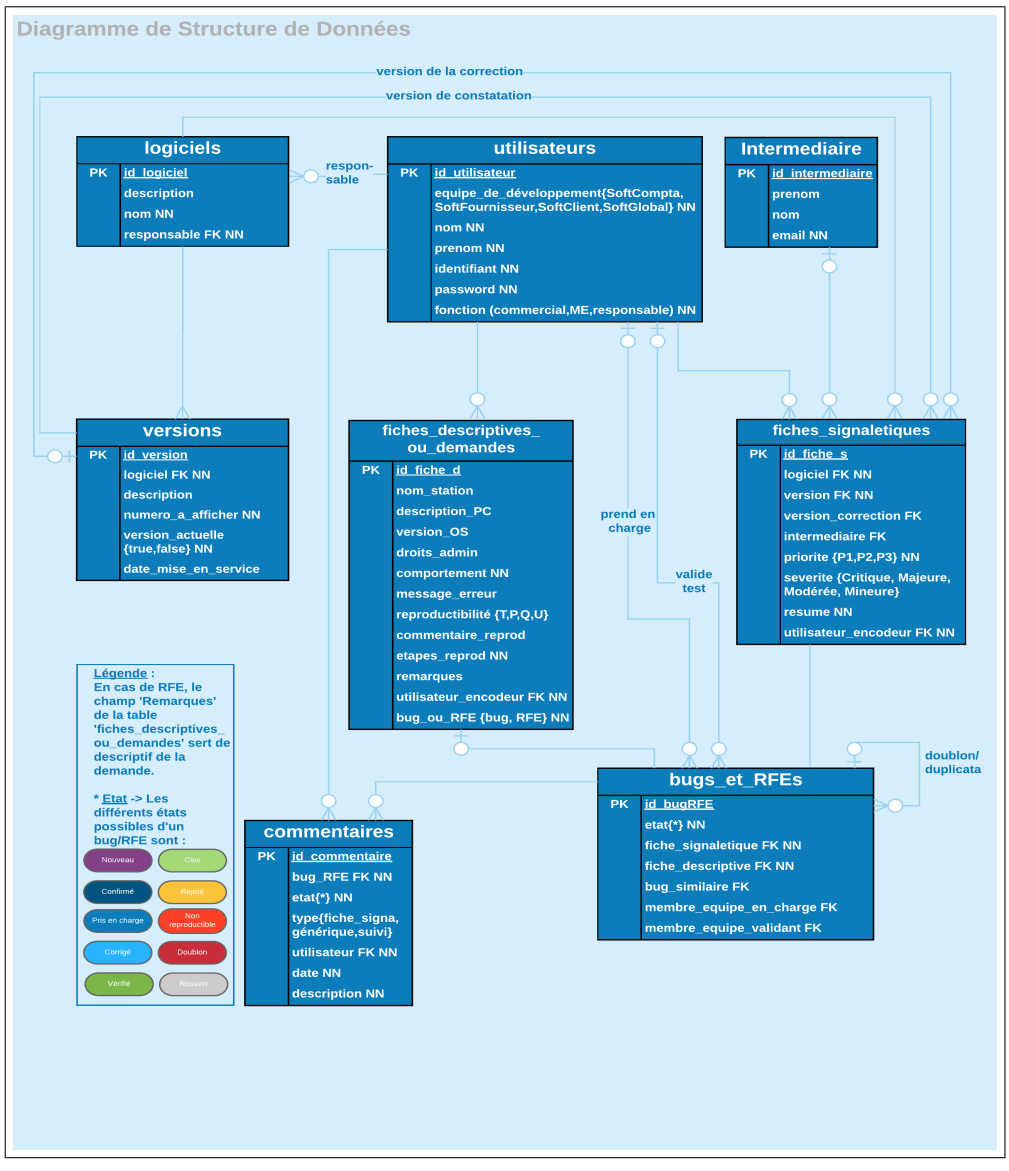

FIGURE 3 – Diagramme de structures des données

Les utilisateurs (commerciaux, membres d'équipe ou responsables) sont encodés dans le système par un responsable grâce à un *fichier csv*. Ils s'identifient en suite dans l'application grâce çà leur identifiant personnel et leur mot de

passe (hashé avec une clé BCRYPT par exemple). L'identifiant et le mot de passe peuvent ensuite être modifiés depuis l'espace personnel de chaque utilisateur.

Les bugs et rfes Les bugs qui sont en état « corrigé »et qui attendent doivent être validés par un membre d'équipe sont dans une version faite par le responsable dans l'environnement de tests externes. Si le bug ne passe pas les tests à ce moment et passe en « rouvert », celui-ci perd également sa version puisqu'il n'est pas pas apte à permettre de passer en production. Celui-ci peut être ensuite recorrigé et revérifier mais peut être assigné à une autre version que la précédente si cette dernière est déjà mise en production.

Les bugs qui ont été corrigé par un membre d'équipe ne sont pas spécialement validés par la même personne, d'où la présence de deux relation entre bug / RFE et utilisateurs. En cas d'échec aux tests, si un bug doit être rouvert, notre application ne prévoit pas de garder un historique de toutes les personnes qui ont pris en charge et corrigé un bug, mais seulement le dernier d'entre eux.

En cas de demande d'amélioration, seul le champs « remarque »de la fiche descriptive est utilisée, c'est ce qui explique la présence de NULL (même pour le champs comportement) dans cette table.

Il est possible de référencer le doublon lié ou le bug initial en cas de bug clos que l'on souhaite rouvrir (FK bug similaire).

L'intermédiaire Il n'est pas un utilisateur de l'application, sa présence dans la base de donnée permet simplement d'envoyer rapidement un récapitulatif de l'historique d'évolution d'un bug directement par e-mail.

Les commentaires jouent à la fois le rôle de commentaire d'une version ou d'un bug / RFE (c'est pourquoi le champ état est NULL notamment) Le champ « état »permet de conserver l'état du bug auquel le commentaire est lié lors de son insertion dans la base de donnée.

5 Glossaire

TO DO