# À propos de l'éditeur

Cet article est publié par le magazine Le train de 13h37.
Le train de 13h37 est un magazine en ligne francophone dédié à la conception Web qui publie régulièrement des articles exclusifs rédigés par des auteurs invités et rémunérés.

Nous avons fait le choix d'une publication gratuite sur notre site, sous licence Creative Commons BY-NC-SA et non financée par la publicité.

Nous éditons aussi des livres papiers et numériques, avec les Éditions En Voiture Simone.

Magazine édité par la société WAGON 42 SAS, au capital de 5 000€ - SIRET : 752 205 856 000 12
25 Impasse Ernest Feuillet - 84000 Avignon // 09.81.61.08.07 – contact@wagon42.fr

# Concevoir un service de rapport de bug

Vous venez de livrer un produit.

Bien évidemment il a des bugs, bien évidemment toutes les spécifications attendues par le client ne correspondent pas exactement à ce qu'il souhaite, bien évidemment le cahier des charges n'avait pas prévu certains cas, bien évidemment il faudra le faire évoluer…

Dès les premières montées en charge, il va vite falloir songer à mettre en place un service de rapport de bugs. Et le construire peut vous amener à mieux comprendre vos utilisateurs.

## *Quel intérêt à construire son propre service de bug report ?*

Les raisons de ne pas en construire un sont légion :

- il existe de nombreuses bibliothèques ou services de feedback prêts à l'emploi ;
- il est pénible de travailler un accessoire purement utilitaire au lieu de se concentrer sur la valeur métier ;
- et d'un point de vue ingénierie, il n'est pas du tout recommandé de faire des couplages forts entre des services qui pourraient être indépendants.

Néanmoins il y a quatre intérêts principaux à construire son propre service de rapports de bugs :
1 - Si vous construisez un backoffice et que celui-ci est destiné à intégrer un service de rapport de bugs, il est très probablement destiné à n'être accessible qu'en `https://`. Utiliser des bibliothèques Javascript qui sont hébergées sur des serveurs tiers que vous ne gérez pas est un risque non-négligeable de sécurité et/ou de confidentialité.
Effectivement, rien ne vous dit que la bibliothèque n'en profite pas pour capturer des données sensibles, ou pire, que le serveur de votre prestataire ne se fasse pirater et que cette bibliothèque soit remplacée par un mouchard. Imaginez toutes les possibilités de profit que constituent les adresses e-mails, les adresses postales et les coordonnées bancaires que vous affichez dans un backoffice.

2 - Il s'agit d'un exercice de base. À mon sens, il est toujours utile de répéter la création de certaines fonctions, comme la gestion login/rescue/logout. Un service de rapport de bugs est aussi pour moi un entraînement : ces exercices permettent de tester si la stratégie de développement (notamment langage et framework) est maîtrisée.
C'est aussi une bonne occasion pour expliquer la philosophie de développement au reste de l'équipe, surtout si le niveau est hétérogène. Il permet aussi de réfléchir à la logique de l'UI avec le webdesigner.

3 - Le formulaire de rapport de bugs doit pouvoir fonctionner même si le Javascript est planté côté client. Cela peut sembler évident, mais je me suis retrouvé coincé sur un produit que j'utilisais à cause de ça, et dans l'impossibilité de faire remonter le problème alors que je pouvais en décrire les causes. Or, la plupart des solutions externes de rapport de bugs fonctionnent via un appel Javascript, comme le très populaire et très complet UserVoice.
L'avantage de passer par un serveur extérieur, c'est que si le serveur de votre applicatif est complètement en panne, ce service annexe pourra toujours enregistrer le retour de "bug" même dans une situation aussi critique.

4 - Il est encore plus facile d'étendre les possibilités offertes par votre service. Car bien évidemment, au fur et à mesure de son utilisation, vous serez tentés de le faire évoluer, tout comme le site qu'il supporte. Parmi les évolutions que vous pourriez envisager (au-delà d'un simple formulaire), songez que vous pourriez enregistrer des événements internes, les données des précédents formulaires de session, pinger un autre webservice… nous verrons plus loin pourquoi.

## *Arriver à le faire accepter*

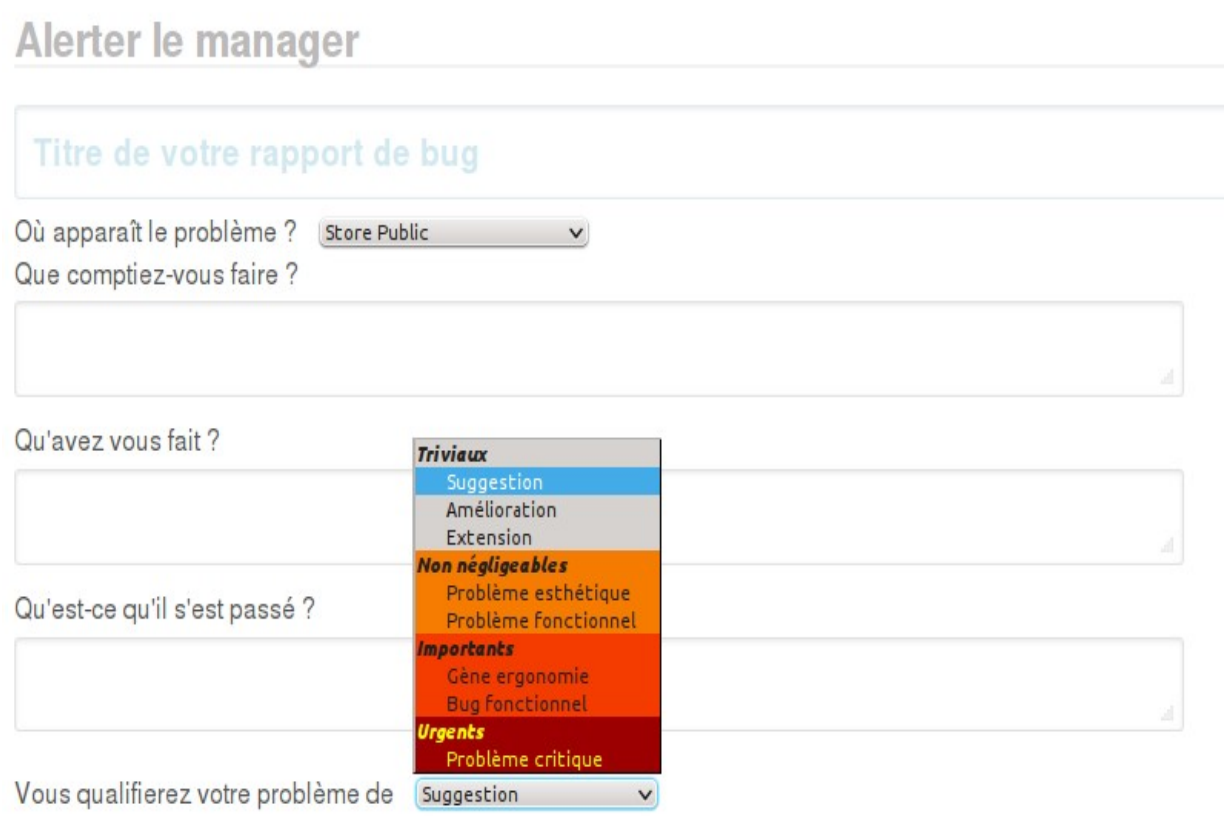
Bien évidemment, de prime abord, votre client ne le verra pas du même œil que vous. D'abord, avoir un service de rapport de bugs, c'est avouer à demi-mot que vous n'avez pas tout fait dans les règles de l'art, et ça, c'est quand même difficile à avaler. Bien sûr qu'il s'en doutait, mais au lieu d'avoir à écrire un texte, vous appeler est facile : dès que vous décrochez, vous êtes immédiatement disponible, peu importe ce que vous faisiez à l'instant. Et le coup de fil lui permet de passer ses nerfs verbalement.
Oui, il y a des urgences, mais votre client est-il en droit de décider à votre place de vos priorités ? Et surtout comment garder trace de ses interventions, des décisions prises, ou de signaler que le "bug" a déjà été signalé et à tort ?
Combien de fois ai-je dû répondre à un "bug" identique qui pourtant a été voulu par le client, car validé dans son cahier des charges.
C'est là-dessus que le travail pédagogique a porté : voir le service de rapport de bugs comme une base de connaissance. Rechercher si le problème en question n'a pas déjà été soulevé, voir les propositions de résolution faites, ou pouvoir me rappeler à l'ordre si une correction prioritaire n'a pas été faite dans les temps.

## *Comment créer un formulaire de rapport de bug ?*

Le formulaire doit aider le plus possible à cerner le problème du premier coup. La règle des 5 W, héritée du journalisme, devrait vous y aider : « Where, When, Who, What, Why ».
Si vous avez fait un peu de Philosophie (si si, ce fameux cours qui permettait de dormir incognito en Terminale), vous devriez relire le chapitre consacré à la Causalité, cela devrait vous donner quelques idées.
Posez à votre interlocuteur les questions auxquelles il pourra facilement répondre. Exercice d'écriture qui, je le reconnais, est relativement compliqué. Demandez ce que l'utilisateur peut décrire. Le navigateur client, c'est inutile : vous pouvez le deviner à son user-agent ; les paramètres passés, c'est là aussi possible de les enregistrer, mais demander à votre visiteur ce qu'il comptait faire est nettement plus utile.
Pour ceux qui se souviennent de la première version du Bugzilla, il était franchement intimidant : il y avait beaucoup trop d'options.
Le but du jeu est d'obtenir un « quand je fais ça, ça fait ça, et pas ça » au lieu d'un « ça marche pas, faut régler ça avant midi ».

Sur dagence, un service que j'avais co-créé, j'avais cherché les questions les plus simples, dans un ordre basé sur la chronologie :

- L'intention : « Que comptiez-vous faire ? »
- Les gestes : « Qu'avez-vous fait ? »
- Le résultat : « Qu'est-ce qu'il s'est passé ? »




Les autres informations étant logguées, notamment les valeurs du formulaire précédemment validé, il était nettement plus rapide à remplir. Mais attention quand vous enregistrez ce genre d'informations, car vous pourrriez mettre à mal les principes de base de sécurité en enregistrant les mots de passe en clair, ce qui serait extrêmement dangereux… Rien n'aurait été plus ridicule que d'appliquer les recommandations minimales de sécurité en hashant un mot de passe, tout en enregistrant à côté le formulaire avec ce même mot de passe lisible en clair comme du plain-text.
Je m'étais donc imposé des "mots clé" qui ne devaient pas être loggués comme tous les paramètres d'un formulaire dont la clé commence par "password_".
Vérifiez toujours que vos journaux de logs n'enregistrent que ce qui est nécessaire.
Petite astuce qui nous avait bien aidé : ce n'était pas le serveur qui stockait ces données. Eh oui, les technologies dites "HTML5" nous donnent plein possibilités.

## *Estimez l'urgence*

Faut-il laisser les utilisateurs décider d'une priorité ?
Devant l'avalanche des "Urgent", "Top prioritaire" et "Ne peut attendre midi", j'avais songé à limiter les possibilités qui lancent une quantité considérable d'alarme chez moi (entre un envoi de SMS et un message sur répondeur, vous voyez le style…). Car c'est quand le souci est réellement majeur que vous ne le voyez pas immédiatement.
J'avais également songé un moment mettre un captcha quand le bug proposé était en mode "Urgent". Mais ce n'était pas idéal car ledit captcha ajoute forcément de la frustration chez la

personne victime du bug.

Étant mainteneur de ma solution en travailleur indépendant, le temps de résolution de faux bugs me posait un sérieux souci de rentabilité. Je cherchais donc à pénaliser les faux bugs sciemment rapportés. En général, des fonctionnalités non demandées à la commande.
Pour contourner ce problème, il y a une technique que je n'ai jamais pu mettre en place, mais qui ne devrait pas être inintéressante : le jeton. Lors de la commande, un certain nombre de jetons sont crédités à votre client.
Si ça vient de moi, la résolution du problème est offerte.
Si le client demande une fonctionnalité qu'il n'a jamais exprimé ou qu'il a retiré de sa commande, il suffit de lui envoyer un devis pour lui rappeller que notre relation commerciale doit être équitable. Si cette fonctionnalité est vraiment utile, il devrait vous retourner le devis signé.

## *Facilitez le dialogue*

Arriver à obtenir un seul bug par rapport est très difficile. J'ai un bel exemple récent où en 8 lignes, on m'en rapporte 10 d'un coup (le produit était en construction).
Prendre un peu de temps à aider le rapporteur de bug, c'est espérer en gagner à l'avenir. Mais montrer qu'on corrige bien plus vite les bugs par ce biais est déjà une première étape. Ajouter un système de suivi, c'est à dire indiquer les étapes de résolution fait aussi partie de cette "récompense" : elle aide à faire comprendre l'attente. Elle montre aussi que vous avez de réelles qualités d'écoutes, et si jamais le bug est très complexe à répliquer, ou qu'il est très difficile à résoudre, le dialogue avec la personne impactée aidera grandement à acquérir plus d'informations utiles ou tout du moins à ne pas lui laisser une impression de dédain.
Le service de bug ne doit pas être un mur, mais une assistance au dialogue. Si vous y montrez des signes d'humilité, vous aiderez d'autant mieux à apaiser les situations tendues.
Vu son effort et le changement dans ses habitudes, bien sûr que le rapporteur s'attend à avoir son bug corrigé au plus vite, ou sinon expliqué. Mais il doit aussi sentir que les réponses sont mieux calibrées et que le dialogue est plus qualitatif par ce biais.
Sur certains projets, j'insiste sur le fait qu'il est une source de bonnes pratiques clients, et qu'il permet aussi à tout usager de pouvoir dialoguer sans avoir à sytématiquement redonner son adresse e-mail. Des fois que, muni de votre adresse e-mail ou de votre numéro de portable, le même usager serait tenté de vous recontacter pour des problème strictement liés à son environnement de travail (ex: un problème d'antivirus, de barre de pub intempestive, d'un Internet Explorer trop vieux…). L'avantage indéniable étant que vous n'aurez à résoudre que ce qui vous concerne réellement.

## *Une bouée de sauvetage*

Le rapport de bugs a un double bénéfice : votre utilisateur doit gagner en confort en aidant à améliorer progressivement le service qu'il utilise, et de votre côté, il doit permettre de tracer vos erreurs les plus fréquentes (ou d'avoir une idée des fonctionnalités les plus demandées).
Intégrer le développement du service de bugs dans votre service est aussi l'occasion de pouvoir placer des métriques, d'analyser les fonctions les plus usitées, et d'estimer celles qui posent le plus de problèmes.
Quand on crée un service Web, on a malheureusement pas la possibilité de faire un logiciel de rapport en cas de plantage, mais avec un peu d'astuce, nous pouvons y palier. Le simple fait de ne pas laisser ses utilisateurs devant une page morte démontre un certain professionnalisme.

Et ça, dans votre relation avec vos utilisateurs, ça fait partir bien des cafards.

Magazine édité par la société WAGON 42 SAS, au capital de 5 000€ - SIRET : 752 205 856 000 12
25 Impasse Ernest Feuillet - 84000 Avignon // 09.81.61.08.07 – contact@wagon42.fr

# À propos de l'auteur

J'ai eu plusieurs vies avant le web : Dresseur de punks à chiens en tant que coordinateur d'une radio associative, puis de concepteur rédacteur dans une entreprise de services téléphoniques surfacturés pour laquelle j'ai lancé les logos et sonneries téléchargeables… Il m'a fallu du temps pour devenir développeur web "professionnel". Durant 7 années en indépendant, je me suis fait une spécialité sur la conception de backoffices. J'ai conçu ma solution qui n'a évidemment pas décollé (dagence.pro) mais qui m'a permis d'expérimenter HTML5. J'ai rejoins Simtie.fr, un service de gestion de documents où je fais le pari qu'un respect des standards aide à écrire pour le très long terme.
Mon blog reste moche, je chronique de la BD et je fais toujours de la radio….

- @dascritch
- http://dascritch.com/