Authentification d'utilisateurs sur une interface WEB

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par:

Marc FALCY

Conseiller au travail de Bachelor :

(Michel KUHNE)

Carouge, 14 mai 2012

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre d'informaticien de gestion. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Carouge, le 14 mai 2012

Marc FALCY

Remerciements

Tout d'abord, je souhaite remercier mon directeur de mémoire, le professeur Michel Kuhne qui a bien voulu prendre la charge de m'orienter. Il a été encourageant dans les moments de relâchement et structurant dans les moments de doute. Appuyé d'une main experte par Johann Sievering, qui a véritablement joué le rôle de boussole, j'ai bénéficié de leur aide pour parcourir un chemin satisfaisant.

Ensuite, je voudrais saluer Michel Dériaz pour l'opportunité qu'il m'a offerte de travailler dans un groupe de recherche et développement de l'université de Genève.

Il est particulièrement important de souligner l'aspect décisif qu'ont joué les différents professeurs, assistants et camarades de la haute école de gestion de Genève m'ayant soutenu par leurs connaissances. Rolf Hauri pour son expertise Ajax, Peter Daehne pour ses précieux et multiples conseils, Christian Stettler pour ses avis PHP, Andrei Starkov pour ces conseils de programmation, Alexandre Steigmeier pour ces connaissances des technologies Web et Nemanja Subotić pour son support jQuery.

Pour finir, je remercie Céline pour ses corrections et son soutien.

Sommaire

But

Afin qu'un utilisateur puisse se connecter à un service sur Internet, il devra fournir à un serveur des données sur lui-même. Le serveur pourra alors procéder à une vérification des données. C'est l'authentification. Deux informations sont à utiliser : un identifiant propre à l'utilisateur ainsi que des règles particulières associées à cet identifiant. Le nom de l'utilisateur comme identifiant ainsi qu'un mot de passe sous forme de suite de caractère sont largement utilisés actuellement. Le but de ce travail est d'étudier un cas particulier d'authentification pour des utilisateurs. Pour ceci, du code a été développé pour l'insérer dans les pages du site *foxyTag*. Ce code doit permettre d'effectuer différentes tâches de gestion d'un profil pour un utilisateur.

Méthode

Le travail a été effectué en utilisant la méthode agile *SCRUM*. C'est une méthode de gestion de projet qui est basée sur l'incrémentation continue du logiciel, ou du produit à fournir au client. Il y a des réunions avec le client toutes les deux semaines afin de lui présenter la production effectuée. Le mandant Monsieur Dériaz a ici joué un double rôle. Celui de client qui s'attend à un logiciel fini, ainsi que de celui de *SCRUM* Master, soit la personne qui rappelle les choix adoptés par les membres de l'équipe, les documents à fournir, les différentes tâches, etc. Afin de pouvoir suivre les cours à la HEG et d'effectuer le travail de Bachelor en parallèle, la meilleure option a été de travailler sur le Bachelor à 40% a été effectué. Le choix de placer le temps attribué au Bachelor du mois de décembre au mois de mai, afin d'être disponible sur le marché du travail déjà pendant la période estivale. En effet, au lieu de réaliser cette étude pendant les mois de juillet et d'août à 100%, ce qui a été le procédé habituel pour les précédents étudiants en informatique de gestion jusqu'à présent, la première option a été décidée.

Avancement

Il a été particulièrement difficile d'avancer alors que la maîtrise des langages de programmation, ainsi que leurs possibilités et limites n'étaient pas connues. Le but premier qui était de mettre en place un système d'authentification et d'enregistrement pour étudier le comportement des utilisateurs qui en feraient l'usage, a dû être remis en cause. Il n'a finalement été possible que de rendre un prototype de cette application.

Table des matières

1. Contexte de développement

Le page développée pour l'enregistrement des utilisateurs pour le site *foxyTag.com* présente quelques lacunes d'ergonomie. Un nouvel outil plus convivial a été demandé par le mandant.

L'outil fourni pour effectuer la validation des données d'un utilisateur au début de ce travail permet une validation des données uniquement du côté serveur. Précédemment, un accès au serveur était donc toujours réalisé. Afin de réduire le nombre d'accès au serveur uniquement aux requêtes d'insertion de données valides, un moteur qui s'occupe de la validation des données de l'utilisateur du côté client a été paramétré. L'intérêt particulier de cet outil est que l'utilisateur puisse directement voir quelles sont les règles de validation de ses données sans avoir à envoyer de requêtes au serveur. De plus, la requête ne sera pas envoyée si ses données sont invalides.

Le test de la page de base de foxyTag, montre qu'il est nécessaire d'effectuer entre trois et quatre connexion au serveur pour fournir des données valides (Figure [1] : Tentatives pour insertion de données valides).

Il est nécessaire de pouvoir effectuer la récupération des données d'un utilisateur au moins une fois. En effet, lorsqu'un utilisateur veut rentrer ses données, il lui faut de toute façon effectuer une requête au serveur afin de savoir si le nom d'utilisateur, son email ou son numéro de téléphone sont déjà utilisés. Avec le moteur de validation mis en place, bien que la connexion au serveur se fasse au moins une fois pour la vérification des données uniques, l'utilisateur n'a pas à changer de page. La convivialité en est particulièrement favorisée grâce aux bulles d'alertes qui viennent se surimprimer à la page affichée par le browser..

2. *SCRUM* comme méthode de gestion de projet

2.1 La thèse et la gestion de projet

Les conséquences de l'agilité sur la manière de gérer cette thèse ont été diverses. Les principaux points positifs ont été de pouvoir adapter les spécifications rapidement grâce à une communication active avec le mandant du projet. En effet, il est favorable d'échanger des points de vue sur l'outil qu'il faut développer et ainsi pouvoir comprendre de mieux en mieux en cours de développement ce qui est dans l'esprit du mandant.

Les principaux points négatifs ont été qu'il n'a pas toujours été facile de s'adapter, ceci étant dû à une évolution des demandes du mandant au fur et à mesure de l'avancement du sujet. Heureusement, les personnes qui ont développé la méthode *SCRUM* ont bien compris qu'il était indispensable de mettre en place des outils pour éviter de tomber dans le même schéma. Bien que ce point soit listé comme point négatif, la méthode mise en place pour la gestion de projet aura été présente pour éviter de refaire les mêmes erreurs. En outre, l'agilité favorise la prise de conscience des erreurs pour constamment s'adapter et corriger les champs étant complétés. Un autre point négatif a été que la version de *SCRUM* utilisée par le groupe de recherche *TaM* dans lequel le travail a été réalisé divergeait de quelques points importants de la méthode étudiée en classe dans le cours du Professeur Dugerdil. Le *burndown chart* a été remplacé par un *burnup chart.* Ceci étant impliqué par le fait que le *planning poker* avait lieu à chaque fin de *sprint* et non pas lieu en tout début de projet. Le poids des tâches n'étant pas défini à l'avance, en tout début de développement, il est donc impossible d'évaluer en cours de développement d'une quelconque vision globale.

Il est à noter que nous avons mis en place un nouveau concept: le "little day". Une journée de travail est composée d'une certaine quantité d'heures dédiées à la production, ainsi qu'une partie dédiée à l'organisation. Nous avons empiriquement constaté qu'il était difficile de bien réaliser les tâches dédiées à la production, que nous nous étions attribuées, si nous ne consacrions aucun temps à l'organisation de la manière à laquelle nous allions réaliser ses tâches. Il a donc été décidé de dédier du temps de la journée de travail à s'organiser en accord avec la méthode *SCRUM*. Ceci doit donc être planifié. La manière qui a été choisie d'adopter pour ceci a été d'effectuer une planification du temps de travail sur un *little day*. Ainsi le *little day* plus le temps quotidien d'organisation équivalent à un jour de travail complet. La mise en œuvre concrète de ceci a pris place lors des *plannings poker*. Ceci nous a permis de prendre conscience du temps à consacrer à l'exécution des tâches d'organisation. Il est particulièrement important de pouvoir s'organiser dans un temps pas trop long, afin que cette tâche ne prenne un temps disproportionné en comparaison du temps de production.

Les conséquences de l'itération incrémentale sur la gestion de la thèse ont produit différents aspects. Le principal aspect positif a été la compréhension rapide des développements à effectuer une fois qu'un point avait été validé. Ceci favorisant la vision globale de l'avancée du projet en cours de réalisation. Le principal aspect négatif a été que bien qu'ayant une vision globale de la tâche à effectuer, les détails n'étaient pas vus a priori. Il ressort de ceci qu'une nouvelle technologie devait être intégrée pratiquement à chaque étape, et que malheureusement il était totalement impossible de le prévoir à l'étape précédente. Un autre aspect, bien que positif pour la connaissance, ne l'a pas été pour la gestion de projet: la formation aux langages utilisés. A chacune des étapes de *SCRUM* où une nouvelle technologie devait être intégrée, il fallait rajouter un temps de formation. Constamment les échéances pour se former ont été repoussées. La conclusion qui est à en tirer est que la méthode de projet qui a été utilisée est fortement compromise à partir du moment où il faut mettre en pause le développement afin de pouvoir mettre en place du temps pour augmenter ses connaissances. En effet, il est toujours impossible a priori d'évaluer le temps nécessaire à la formation sur un outil ou un langage que l'on ne connait pas. Il aurait été ici intéressant de pouvoir utiliser le *burndown chart* fourni par le système de gestion de *SCRUM*, afin de pouvoir évaluer le ratio entre le temps de formation et le temps de réalisation du mandat grâce à l'élévation du temps planifié du projet. Malheureusement, la pertinence de ce graphique est faible en présence d'ajouts constants de tâches. Dans le *burnup chart* de fin de projet (Figure [2] : Burnup chart), les points de stagnation sont les moments de formation à une nouvelle technologie demandant un ajout de temps particulier.

Il est ainsi justifié de prendre en considération que la formation prend un temps important sur la production. L'idéal serait de retirer les développeurs qui doivent se former du processus de gestion de projet et les réintégrer une fois qu'ils sont formés. Dans ce cas, la méthode *SCRUM* est parfaite pour le développement de logiciel en entreprise, car il n'y a que relativement très peu de formation. Elle est moyennement efficace pour la recherche, car la formation peut se faire en un temps qui n'est pas limité. Elle est finalement mauvaise dans le domaine de l'étude, car la formation doit se faire en un temps qui est défini d'avance.

Le double rôle de Michel Dériaz en tant que *Product Owner* ainsi qu'en tant que *SCRUM* Master a été particulièrement bénéfique pour l'utilisation de ses connaissances. Ne disposer que d'un seul interlocuteur est utile si des questions sur le déroulement du *SCRUM* ou sur le produit à développer se présentent. Il a souvent pu orienter les réponses qu'il faisait pour l'élaboration du produit afin que le développement puisse être en accord avec la méthodologie de *SCRUM*.

3. Les différents langages de développement

Le langage dans lequel il était prévu de réaliser le prototype était au tout début du projet le HTML5. Bien que ce langage soit extrêmement utile dans le domaine des formulaires pour leur validation, il a été révélé en cours de réalisation que des particularités lui manquaient pour remplir les termes du mandat.

En effet, il a été nécessaire d'envisager un moteur de validation utilisant du JavaScript. Il est apparu que bien que le moteur de validation du HTML5 ne soit pas limité, ses fonctions dynamiques risquaient de ne pas être acceptées par de vieux navigateurs. De plus, l'appel des procédures de validation pour le formulaire de HTML5 ne se fait que lorsque l'utilisateur clique sur le bouton "Envoyer le formulaire", un moteur faisant la validation à la volée étant plus approprié pour le mandant, le choix s'est étendu au JavaScript et au CSS dans lequel le moteur était construit[1].

Etant donné que l'enregistrement d'un utilisateur devait être géré, ainsi que la possibilité de celui-ci à modifier après coup ses données, la situation cadrait particulièrement avec une session d'utilisateur. Pour ceci deux possibilités ont été comparées. L'utilisation de cookies et l'utilisation de variables de session. Il a par conséquent fallu faire le choix d'étendre la liste des langages utilisés au PHP. Leur comparaison par matrice de préférence a déterminé le choix des variables de session (Figure [3] : matrice de préférence, cookies et version). En effet, les variables de sessions répondent à 90% aux critères secondaires (les critères primaires étant indispensables), les cookies présentant un sérieux désavantage, à savoir le fait que certains browsers les refusent. Les variables de session, en plus de ne pas souffrir de ce désavantage, bénéficient d'un point positif. Elles sont entièrement gérées du côté serveur.

Lors de la réalisation du prototype, des pages d'accès à la base de données ont été utilisées pour valider les données rentrées par l'utilisateur[2]. Le *PHP* permet cette possibilité. Cela réalisé, et afin d'augmenter la sécurité, le mandant requérant de ne pas laisser dans le code le nom d'utilisateur et le mot de passe pour l'accès à la base de données, il a donc fallu envisager l'utilisation du langage Ajax pour réaliser les requêtes au serveur grâce aux requêtes asynchrones que propose ce langage. Ceci directement depuis les pages d'insertion des données[3].

---

[1] Il a donc été nécessaire d'intégrer une formation express au JavaScript et au CSS.

[2] Cf. : Annexe 7.4.1

[3] Cf. : Annexe 7.4.2

3.1 Partie *HTML*

La première idée a été d'utiliser le langage *HTML*5 pour écrire les pages pour le site. Le but étant d'utiliser ses fonctionnalités pour formulaires.

Malheureusement, le *HTML*5 n'est pas suffisamment développé pour les plateformes mobiles au moment du début du travail. Pour cette raison et pour le fait que le *HTML*5 n'était pas indispensable pour la réalisation de la validation des champs remplis par l'utilisateur, le mandant à fait le choix de rester dans la version précédente de *HTML*.

Il est à noter que pour des raisons de fonctionnalité dans un futur de l'application, le langage *HTML*5 serait probablement plus utile que la version actuelle. Par contre, il sera toujours nécessaire d'utiliser un langage pouvant faire des requêtes asynchrones à un serveur[4].

Le *HTML* en devient indispensable pour l'utilisation de formulaires, ainsi que de l'insertion des champs nécessaires à ceux-ci.

3.2 Partie *JavaScript*

Le *javaScript* est employé pour ce travail afin d'effectuer la validation des données insérées par l'utilisateur.

Pour réaliser ceci, deux possibilités étaient disponible. Soit créer les fonctions *javaScript* de toute pièce. Soit utiliser un moteur de validation. Parmi les outils disponibles, deux moteurs ont été testés. C'est le choix du plugin jQuery qui a été retenu (Figure [4] : Matrice de préférence, validation *javaScript*). Le choix du moteur a été favorisé par la littérature, ainsi que les nombreux exemples le concernant. C'est un moteur écrit en JavaScript qui utilise la bibliothèque *jQuery* pour effectuer son travail. Il faut pour ceci appeler le moteur de validation depuis la balise input du code HTML que l'on veut contrôler. Si la validation n'est pas effectuée, le moteur se chargera d'afficher un message d'alerte et empêchera l'envoi du formulaire. Si la validation est effectuée, le moteur supprimera les messages d'alerte et permettra l'envoi du formulaire. On insérera les fonctions à utiliser pour la vérification. Ces fonctions sont modifiables dans le moteur de validation en les renommant ou en changeant leur expression rationnelle attribuée. Il est ainsi possible de modifier le moteur afin de l'ajuster à sa guise[5].

[4] La justification est au paragraphe Ajax.

[5] Cf. Annexe 7.4.4

Ci-dessous sont présents deux exemples d'appel du moteur de validation. Depuis les champs du *username* et du *password* (Figure [5] : Appel du moteur de validation). Un exemple de modification des fonctions appelées dans le moteur de validation (Figure [6] : Modification des règles validantes).

Le moteur de validation est appelé lorsque dans un champ concerné, il y a un clic de souris, l'acquisition du focus, un relâchement d'une touche ou quand le texte a été changé. Il est important que chacun de ces événements provoquent l'appel du moteur, puisque il y a différentes manières de modifier un champ. Ecrire dans le champ sera repéré par "keyup" alors que effectuer un coller dans le champ sera repéré par un "change". Voir l'exemple de modification des paramètres de lancement du moteur de validation (Figure [7] : Différents comportements de l'utilisateur).

Sur les pages de login[6], d'enregistrement[7], de modification des données de l'utilisateur connecté[8] ou de demande d'un nouveau mot de passe[9], deux directions sont disponible. Ces directions sont fonction de l'avancée du travail. La première direction sera d'analyser le comportement de l'utilisateur et de donner le traitement à réaliser aux pages de contrôle en *PHP*[10]. La seconde direction sera de traiter le comportement de l'utilisateur et de poursuivre le traitement avec des requêtes Ajax[11], suivies de fonctions JavaScript utiles au traitement des données en retours du serveur oxgva. Quelle que soit la direction empruntée, le comportement de l'utilisateur à repérer est le même.

Dans les spécifications du mandat, il est demandé que le bouton d'envoi des données ne soit utilisable que lorsque toutes les données auront été insérées correctement. Ici l'état du bouton d'envoi doit être changé si les données sont valides. Pour une question de temps, ce point n'a pas pu être réalisé.

---

[6] loginPage. Cf. Annexes 7.4.2

[7] registrationPage. Cf. Annexes 7.4.2

[8] updateDataPage. Cf. Annexes 7.4.2

[9] passwordForgottenPage. Cf. Annexes 7.4.2

[10] Pour le traitement *PHP*, voir le traitement référent au chapitre concernant le *PHP*

[11] Pour le traitement Ajax, voir le traitement référent au chapitre concernant l'*Ajax*

3.2.1 Scénario *PHP*[12]

Sur la page de login, lorsque l'utilisateur fait l'action de se loger, son *username* et mot de passe sont transférés à une page de contrôle[13] par l'envoi d'un formulaire. Cette page de contrôle se chargera de vérifier si l'utilisateur est bien présent dans la base de données du serveur et s'occupera de faire la redirection opportune. Sur la page d'enregistrement d'un utilisateur, en plus des *username* et *password*, comme pour la page de login, il y aura en plus les chaînes de caractères correspondant à la langue, au email et au numéro de téléphone de l'utilisateur qui seront transférés à la page de contrôle correspondante[14]. Sur la page de modification des données d'un utilisateur, du code *javaScript* est utilisé pour insérer les nouvelles données dans des champs cachés du formulaire. A côté des anciennes données elles pourront être envoyées à la page de contrôle correspondante[15] qui se chargera des redirections en fonction des champs modifiés. Sur la page de demande d'un nouveau mot de passe, ce n'est que le mail qui est transféré[16].

3.2.2 Scénario Ajax[17]

Sur la page de login lorsque l'utilisateur clique sur le bouton de login, la fonction checkData[18] va être appelée. Pour cette situation, cette dernière va effectuer plusieurs requêtes Ajax. Leurs retours permettront de choisir de loguer l'utilisateur en le dirigeant sur userPage, ou de lui demander d'insérer des données valides en restant à la même adresse.

Sur la page d'enregistrement d'un utilisateur. La fonction checkData se charge d'envoyer la requête Ajax permettant de vérifier que les données uniques insérées par l'utilisateur ne sont pas déjà utilisées par quelqu'un d'autre. Si elles ne sont pas utilisées, une nouvelle requête d'enregistrement sera envoyée et le nouvel adhérent sera dirigé sur une page de préconfirmation. Si elles sont déjà insérées par quelqu'un d'autre, la même page sera affichée avec les messages d'alerte correspondants.

---

[12] Cf. partie 3.5. La Figure [21] : structure des pages du site, avec pages de contrôle est utile à la compréhension de la structure

[13] loginControlPage. Cf. Annexes 7.4.1

[14] registrationControlPage. Cf. Annexes 7.4.1

[15] updateControlePage. Cf. Annexes 7.4.1

[16] passwordForgottenControlPage. Cf. Annexes 7.4.1

[17] Cf. partie 3.6. La Figure [22] : structure des pages du site, avec requêtes *Ajax* est utile à la compréhension de la structure

[18] onsubmit="return checkData()"

Sur la page de modification des données d'un utilisateur (Figure [8] : Page d'enregistrement), il sera vérifié les différentes données modifiées grâce à des champs cachés. Plusieurs champs sont modifiables: Le nom de l'utilisateur, son adresse email, son numéro de téléphone portable, son mot de passe et sa demande de confirmation, ainsi que la langue de notification. La fonction checkData se charge de regarder si un ou plusieurs des champs uniques (*username*, *mail*, *phone*) ou le mot de passe ont été changés. En ce cas, c'est le scénario « updateCriticalData ». Si aucun de ces précédents champs n'ont été modifiés, mais que le champ langue l'a été, nous entrons dans le scénario « updateLanguage ». Une autre possibilité est d'avoir une fusion des deux précédents scénarios pour donner le troisième scénario appelé « updateCriticalData + Language ». En fonction des changements effectués, différents appels Ajax peuvent être réalisés, chacun redéfinissant l'adresse à suivre pour le comportement correspondant à la réponse. Soit sa page spécifique en cas de réussite ou le rappel de la page en cours, avec le message d'alerte correspondant en cas de retour négatif.

Sur la page de demande d'un nouveau mot de passe la requête Ajax permettra de définir si un mail a été envoyé ou non. Si c'est le cas, la suite se dirigera vers la confirmation de l'envoi d'un mail.

3.3 Les expressions rationnelles

Les expressions rationnelles utilisées pour valider les différents champs font partie des règles de validation du moteur. Bien qu'elles soient partie prenante du document "jquery.validationEngine.js", il faut les distinguer des autres sous-chapitres. En effet, les *expressions rationnelles* font partie d'un langage en soit et sont clairement distinctes du JavaScript. Elles sont soit reprises tel quel parmi celles fournies par le mandant, celles-ci venant du système de validation du serveur, soit refondues pour ce travail pour en comprendre le fonctionnement. Le champ du nom de l'utilisateur *username* est validé par les règles "required", "custom[loginName]" et "custom[loginSpace]". Il aurait été possible de ne réaliser qu'une seule règle pour la validation de ce champ, mais il est utile de diviser les expressions pour simplifier leur compréhension et pouvoir en réutiliser une partie si nécessaire. "required" définit que le champ est indispensable. Il est à noter que la définition de cette expression est vide (Figure [9] : Expression rationnelle de required), ceci pour réaliser le code de la fonction du même nom que celle-ci étend (Figure [10] : Expression rationnelle étendue).

"custom[loginName]" définit que le champ peut avoir entre trois et vingt-cinq caractères spécifiques (Figure [11] : Expression rationnelle de loginName).

"custom[loginSpace]" définit que la chaîne de caractères peut comporter un caractère espace à plusieurs places dans la chaîne à condition qu'il ne se trouve ni en première

position, ni en dernière et que deux espaces ne soient pas côte à côte. Il y a ici une division possible de la formule en une partie plus petite. "custom[loginSpace]" qui n'a finalement pas été utilisée vérifiait qu'il n'y ait pas d'espace en début ou en fin de chaîne (Figure [12] : Expression rationnelle des loginSpace).

Le champ *mail* de l'utilisateur est validé par les règles "required" et "custom[email]". "custom[email]" définit que la chaîne de caractères doit correspondre à un e-mail valide (Figure [13] : Expression rationnelle de mail).

Le champ du numéro de téléphone de l'utilisateur *phone* est validé par les règles "required" et "custom[telephone]". "custom[telephone]" définit que la chaîne de caractères doit comporter le symbole "+" et est suivie d'un nombre de huit à seize chiffres. Ceci pour correspondre au format international des numéros de téléphone (Figure [14] : Expression rationnelle *phone*).

Le champ du mot de passe de l'utilisateur *password* est validé par les règles "required" et "length[7,25]". "length[7,25]" définit que la chaîne de caractères doit comporter entre sept et vingt-cinq caractères quelconques. Elle étend aussi une autre fonction (Figure [15] : Expression rationnelle de longueur).

Le champ de confirmation du mot de passe de l'utilisateur (confimPassword) est validé par les règles "required" et "confirm[password]". "confirm[password]" définit que la chaîne de caractères doit être la même que celle du mot de passe (Figure [16] : Expression rationnelle de confirmation de mot de passe).

3.4 Partie *CSS*

Le message d'alerte de validation du moteur est activé par divers événements décrits dans la partie JavaScript. Il est important de noter que le texte du message d'alerte est récupéré dans les règles du moteur de validation présentées au paragraphe des expressions rationnelles. Si l'on veut changer les textes référents, il faut donc modifier les règles du document écrit en JavaScript. Le message d'alerte produit par le moteur de validation est ensuite affiché à côté du champ correspondant de la page (Figure [17] : affichage des messages d'alerte).

Pour ce qui est des spécificités particulières du message en soit, soit couleur et taille, il faut se reporter au document "validationEngin.jquery.css". Il est possible de modifier la position et la taille de la boîte d'alerte en modifiant les paramètres de la fonction ".formErrorContent" (Figure [18] : position et la taille de la boîte d'alerte). Il aussi possible de modifier la position de la flèche de la boîte d'alerte en modifiant les paramètres de la fonction ".formErrorArrow" (Figure [19] : position de la flèche de la boîte d'alerte), ainsi

que sa taille en attribuant des lignes superposées à cette fonction (Figure [20] : taille de la flèche de la boîte d'alerte).

3.5 Partie PHP

Il a fallu effectuer l'interrogation de la base de données des utilisateurs de foxytag pour les différentes raisons suivantes[19]. En fonction du contenu de la réponse à la requête, l'utilisateur est dirigé sur la page appropriée suivante. Cette structure complexe de pages (Figure [21] : structure des pages du site) a été mis en place alors que le langage Ajax n'était pas encore utilisé pour le développement. Comme l'indiqué dans la partie consacrée à Ajax, ce langage permet de supplanter tout ce lourd mécanisme de pages mises en place.

Depuis la page de login, un accès à la page de contrôle "LoginControlPage" est effectué. Cette page sert à vérifier que l'utilisateur possède un login et qu'il fournit le mot de passe correspondant à son login. Ceci doit être réalisé quand celui-ci veut se loger. Si le login est autorisé, la page suivante est la page de l'utilisateur (userPage), alors que si le login n'est pas autorisé, il y a un retour sur la page de login et un message d'alerte indiquant une erreur de *username* ou de mot de passe.

Depuis la page d'enregistrement d'un utilisateur, un accès à la page de contrôle "RegistrationControlPage" est effectué. Cette page sert à vérifier que les données que l'utilisateur insère n'existent pas encore. Ceci doit être réalisé quand celui-ci veut s'enregistrer. Si les données n'existent pas encore, lors du changement de page les données sont transférées et la page suivante est celle qui confirme à l'utilisateur qu'il a inséré les données et qu'il doit accéder à son mail pour effectuer leur confirmation (registrationConfirmationPage). Si le *username*, le mail ou le numéro de téléphone étant insérés existent déjà, la page suivante sera la page informant l'utilisateur de ce fait (respectivement registrationBddUsernamePage, registrationBddMailPage ou registrationBddPhonePage). Si plusieurs des données que l'utilisateur veut insérer sont déjà dans la base de données, il n'y a l'annonce que d'un attribut déjà présent. Ainsi l'utilisateur pourra corriger l'attribut annoncé et effectuer un nouvel essai d'insertion qui lui indiquera l'autre attribut à changer. Si plus de temps devait être disponible pour le projet, une correction de cette lourdeur serait la bienvenue. Il faudrait appliquer des variables de session contenant un texte à afficher à la page d'alerte suivant. Ce texte serait fonction du cas dans lequel l'utilisateur se trouverait. Il n'y aurait ainsi qu'une seule page d'alerte, qui serait nommée registrationBddPage, et il n'y aurait plus ces trois pages différentes actuellement.

[19] Réalisé grâce à l'utilisation des pages intermédiaires de contrôle

Depuis la page de modification des données d'un utilisateur, différents comportements sont possibles. Ces comportements différents que produit l'utilisateur sont repérés par le code JavaScript décrit plus haut. Les conséquences sont que trois différentes pages de confirmation peuvent être accédées. La première page de contrôle à laquelle l'utilisateur peut accéder est "updateControlPage" lorsque le scénario "updateCriticalData" est appliqué. Elle sert à vérifier que les nouvelles valeurs que l'utilisateur insère n'existent pas encore. Ceci doit être réalisé quand celui-ci veut changer ses données uniques (*username*, email, *phone*) ou son mot de passe. Le comportement de cette page ressemble beaucoup au comportement de la page de l'enregistrement de l'utilisateur. Ce qui change, ce sont uniquement les pages de destination. En cas de concordance avec des champs déjà présents dans la base de données, les pages suivantes sont updateBddUsernamePage, updateBddMailPage ou updateBddPhonePage. Si l'unicité des champs remplis est confirmée, lors du changement de page, les données sont transférées au serveur et la page suivante est celle qui confirme à l'utilisateur qu'il a modifié ses données et qu'il doit accéder à son mail pour effectuer leur confirmation (updateConfirmationPage).

La deuxième page de contrôle à laquelle il est possible d'accéder est "UpdateLanguageControlPage". Cette page sert à changer la langue des notifications à l'utilisateur. Attention, ce n'est pas la langue de site, mais la langue dans laquelle l'utilisateur reçoit les mails. Ceci doit être réalisé quand l'utilisateur fait l'action de modifier cette langue dans le scénario "updateLanguage". La page suivante est la page qui annonce le changement effectué, soit "updateLanguageConfirmationPage".

La troisième page de contrôle accessible est "updateLanguageDataControlPage". Elle est directement suivie par "updateDataAfterLanguageControlPage". Elles sont utiles lorsque l'utilisateur veux changer sa langue de notification ainsi que ses données personnelles (scénario "updateCriticalData + Language"). La page suivante est obligatoirement "updateLanguageDataControlPage" qui va réaliser le même travail que "UpdateLanguageControlPage". Par contre la page d'après sera obligatoirement "updateDataAfterLanguageControlPage" qui va effectuer le même travail que la page "updateControlPage". Les problèmes de données déjà présentes pourront être traités de la même manière, vu que la langue aura déjà été changée et que nous entrons ainsi dans le comportement spécifique à la première option de page.

Depuis la page de demande d'un nouveau mot de passe, un accès à la page de contrôle "passwordForgottenControlPage" est possible. Cette page sert à vérifier que l'utilisateur fournit une adresse mail étant dans la base de données. Ceci doit être réalisé quand ce dernier a oublié son mot de passe et qu'il veut s'en faire envoyer un autre à son adresse mail. Si l'adresse mail existe, un mail est envoyé depuis le serveur et la page suivante (passwordForgottenMailedPage) est affichée. Si l'adresse email n'existe pas, il y a un

retour sur la page précédente et un message d'alerte indiquant l'inexistence du mot de passe.

Afin de mettre en œuvre une structure solide de session, les variables doivent transmettre différentes informations entre les pages qui se suivent. Soit les différentes étapes des trois processus particuliers (enregistrement, login ou récupération de mail). Chacun d'entre eux possède une étape ou step qui définit s'il est possible de retourner en arrière. Il est par exemple inutile qu'un utilisateur puisse revenir en arrière une fois qu'il a modifié ses données et reçu le mail de confirmation de leur changement. Il faut donc que l'utilisateur soit déconnecté de sa session et que s'il rappelle la page précédente, celle-ci contrôle l'état des variables et ainsi force l'accès à la page d'accueil.

Plus généralement, lorsqu'une étape est atteinte, une variable de session devra être modifiée en incrémentant son « âge ». Chacune des pages devant toujours faire un contrôle pour ne pas permettre de revenir en arrière lorsqu'un utilisateur a fermé sa session, ou réalisé des comportements sémantiquement irréversibles.

Afin de pouvoir assurer une communication entre deux différents serveurs, une requête Ajax ne peut être effectuée depuis un browser. Il a été constaté que le serveur auquel la requête était faite recevait bien la demande, et faisait bien l'action de répondre. Par contre, c'est entre le moment de l'envoi de cette réponse par le serveur et sa réception par le browser qu'une sécurité mise en place bloque ce retour. Grâce au *wrapper* fourni par Monsieur Hauri, il a été possible de demander au serveur de résidence des pages développées de retransmettre la demande[20]. Chaque différente *requête GET* à envoyer au serveur de base de données contenant un nombre de paramètres ainsi qu'une nomenclature différents, il a fallu réaliser différents *wrappers*[21].

Le *wrapper*[22] permettant l'envoi de deux paramètres (*username* et *password*) servant lors du login à récupérer mail, numéro de téléphone et langue de l'utilisateur. Ainsi ce sont les données récupérées à ce moment qui sont présentées à l'utilisateur comme ses données propres.

Le *wrapper*[23] permettant l'envoi de trois paramètres (*mail*, *phone* et *language*) utile à la vérification de la disponibilité des valeurs entrées par l'utilisateur. Il sert aussi par l'action de la réutilisabilité à retransmettre la requête d'enregistrement d'un utilisateur.

---

[20] Ce sont des requêtes Cross Domaine

[21] Cf. Annexes 7.4.5

[22] AjaxCrossWrapperGetUserDetails

[23] AjaxCrossWrapperCheckRegisterData

Un autre *wrapper*[24] utilisant aussi trois paramètres avec des noms différents (*username*, *password* et *language*) permet de modifier la langue d'un utilisateur. Dans un autre cas de modification des données d'un utilisateur, un *wrapper*[25] contenant un nombre plus élevé de paramètres est utile lorsque des données sensibles, parce que uniques, sont modifiées. A savoir toutes les anciennes données, ainsi que les nouvelles (oldUsername, oldPassword, oldEmail, oldPhone, *username*, *password*, *email*, *phone* et *language*).

Le seul qui n'envoie qu'un seul et unique paramètre est celui qui s'occupe de transférer le mail au serveur afin de pouvoir réaliser une récupération de mot de passe perdu.

3.6 Partie *Ajax*

Dans le contexte de ce travail, l'Ajax sert à supprimer les pages de contrôle et de faire leur travail particulier dans les pages précédent les pages de contrôle (Figure [22] : structure des pages du site, avec requêtes *Ajax*). Ceci grâce à des requêtes Ajax dont les résultats sont traités dans des fonctions JavaScript. Afin d'effectuer ce travail, des requêtes Ajax sont ajoutées au code JavaScript. Ces requêtes permettent d'envoyer les requêtes http avec différents paramètres (Figure [23] : requête Ajax. Récupération de mot de passe) (Figure [24] : requête Ajax. Vérification des données).L'intérêt est que ces requêtes http seront traitées par le serveur en fonction d'une commande y incluse. Cette commande est préfixée par les quatre caractères "cmd=" suivis par le nom de la commande. Les différentes commandes utilisées sont: checkData, register, recover, getEmail, getPhone, getLanguage, updateCriticalData et updateLanguage. Chacune de ces commandes à une fonction particulière utile dans une ou plusieurs pages du site. Etant donné que l'appel de ses requêtes est fait lorsque l'utilisateur a fini de remplir ses champs, autrement dit, quand il valide son choix en appuyant sur le bouton, il faut que ces requêtes soient envoyées de manière synchrone pour en tirer une réponse à un moment précis. Si leur envoi devait être fait plus tôt et qu'un rajout d'un caractère devait être fait, il y aurait une surcharge inutile de communication et une surcharge du serveur puisqu'il y aurait la possibilité de réaliser plusieurs requêtes pour une demande de validation. Pour un travail ultérieur il est à ajouter qu'il ne serait pas possible d'analyser les logs du serveur sans devoir en démêler cette inutile surcharge de données.

Lorsque le langage Ajax a été abordé, un problème particulier est apparu. Il n'était pas possible d'effectuer des requêtes depuis les pages fournies par un serveur sur un domaine qui ne soit pas le même.

---

[24] AjaxCrossWrapperUpdateLanguage

[25] AjaxCrossWrapperUpdateCriticalData

Deux alternatives étaient possibles afin de ne pas être obligé d'utiliser un *wrapper* dans le serveur foxyTag tout en gardant l'architecture des deux serveurs. La première possibilité était d'intégrer une ou plusieurs iFrame dans la page de base. Ceci aurait permis de ne pas utiliser de l'Ajax, mais cette possibilité a été rejetée par le mandant pour la raison que cette méthode n'est plus validée par les normes du W3C. La seconde possibilité aurait été de transférer la tâche de *wrapper* à un service web externe. Le plugin jQuery de John Resig[26] utilise un service web de Yahoo pour effectuer cette possibilité. Bien que cette possibilité soit un parfait exemple de réutilisabilité, il pose de sérieux problèmes de sécurité. Puisque l'envoi des requêtes passe par un service externe, les mots de passes étant traités en clair, il n'est pas envisageable de laisser une entité externe pouvoir disposer des mots de passe des clients du site foxyTag. Cette méthode pose aussi des problèmes de confidentialité des données, puisque des adresses email et des numéros de téléphones sont envoyés à l'application de Yahoo. Il est donc compréhensible que le mandant ait aussi refusé cette possibilité.

L'architecture ne permettant pas cette liberté de communication entre browser et serveur, un plan offrant la possibilité de visualiser cette structure, a été réalisée (Figure [25] : Architecture des serveurs concernés. Structure insuffisante). Le diagramme de séquence a été effectué pour permettre de comprendre l'ordre des opérations (Figure [26] : Séquence diagramme des requêtes). Or, en effectuant cette visualisation, et en la présentant à l'expert Ajax de l'école, Monsieur Rolf Hauri, il est devenu clair qu'une récupération de données d'un serveur par un browser n'était pas possible. Pour éviter des problèmes de sécurité, cette possibilité a été supprimée des options d'*Ajax*. Il est alors devenu indispensable de faire traiter la récupération des requêtes non pas par un browser, mais par un serveur. Ensuite de quoi le serveur se chargera de communiquer les données récupérées au browser. Pour ceci, le code d'un *wrapper* a été fourni par M. Hauri. Ce code a été intégré à l'architecture présente (Figure [27] : Architecture des serveurs concernés. Structure nécessaire) et a pleinement joué son rôle de transfert de communication (Figure [28] : Séquence diagramme des requêtes. Structure nécessaire). Cette solution offre l'avantage de ne pas de ne pas modifier les services déjà mis en place.

[26] https://github.com/padolsey/jQuery-Plugins/tree/master/cross-domain-ajax

4. Les différentes structures

4.1 Les use-cases

Les use-cases tirés des besoins pour la gestion d'utilisateurs (Figure [29] : use-cases) permettent de comprendre l'implication de chaque fonction. Les sous use-cases « vérifier l'absence de données semblables » et « vérifier présence de données valides » sont typiquement mis en jeu par les pages de contrôle et par les fonctions *Ajax,* respectivement pour les scénarios *PHP*[27] et Ajax[28]. Le sous use-case « vérifier validité des données » est spécifiquement utilisé par l'ensemble des pages (à l'exception de userPage qui n'est une présentation des données de l'utilisateur) pour réaliser en une pierre deux coups le contrôle des champs remplis et l'information des règles d'insertion de chaîne de caractère à la personne sur cette page.

4.2 Le mandat

Le mandat est de réaliser une application permettant d'enregistrer un utilisateur, de le loger, de lui permettre de modifier ses données ainsi que de récupérer un nouveau mot de passe par mail si l'utilisateur a oublié le sien. Un guide utilisateur est fourni dans l'annexe

Bien que le temps à disposition n'ait pas été suffisant, il était aussi demandé d'analyser les comportements des utilisateurs de manière à proposer une version du prototype qui encouragerait les utilisateurs à fournir leurs données.

[27] Cf. paragraphe 3.2.1

[28] Cf. paragraphe 3.2.2

5. Le développement

5.1 Les outils

Afin de réaliser les différentes tâches de ce travail, des outils ont été nécessaires. Pour réaliser le code des pages, il a été utilisé la structure déjà existante de la première version de la page d'enregistrement du site *foxyTag*. La structure de chacune des pages développées y sont toutes basées. L'outil Notepad++ a facilité la compréhension et lecture du code, grâce au coloriage du code en fonction des différents langages, fonctions et variables.

Quand du code Php a commencé à être développé, une virtualisation d'un serveur en local a été nécessaire. L'outil Apache WampServer a été installé et utilisé en tant que tel.

MySQL Workbench a été utilisé pour vérifier les insertions à la base de données ainsi que les différentes requêtes HTTP qui étaient faites au serveur. En effet, le serveur possédant une table des différents logs, il était alors possible de vérifier les comportements engendrés sur le serveur oxgva contenant la base de données.

Les browsers utilisés ont été Firefox, avec son outil de programmation firebug, ainsi que chrome.

Le moteur de validation jQuery intégré au code javaScript a permis la validation des fonctionnalités[29]

[29] Source : https://github.com/posabsolute/jQuery-Validation-Engine

6. Les difficultés rencontrés

Les nouvelles technologies ont beaux être compréhensibles facilement, il y a toujours un détail qui vient se rajouter. Ce détail a provoqué un besoin d'accroître les connaissances sur le sujet. Bien qu'il ait été prévu dans la gestion de la réalisation du travail de Bachelor une partie formation, ce temps a complètement été sous-estimé, et il était impensable au départ de devoir faire appel à un si grand nombre de technologies. Pour chacune de ces technologies, du temps a été pris afin de pouvoir les maîtriser suffisamment. La maîtrise devait être à la hauteur de la tâche à réaliser et pour laquelle cette technologie était nécessaire.

Une fois qu'une partie était développée, il fallait constamment garder en conscience de tout ce qui avait été déjà fait pour rajouter une simple modification. La vision globale était nécessaire pour l'intégration d'un point particulier.

Pour le problème Ajax, la principale difficulté a été de comprendre le problème. Il en a résulté une faible réactivité. Ainsi il y a eu une difficulté à résoudre le problème ainsi qu'un retard dans le développement du prototype.

7. Annexes

7.1 SCRUM

7.1.1 L'équipe de projet

L'équipe de projet est composée de trois rôles de personnes. Le *SCRUM* Master, au cours du travail de Bachelor, ce rôle est joué par Michel Dériaz. Il veille à l'application à la lettre des règles de gestion de projet, comme l'élaboration standardisée des documents, ainsi que leur reddition dans les délais.

L'équipe de développement. Bien que dans les standards de SCRUM les équipes sont prévues pour être comprises entre cinq et vingt personnes, nous avons toujours été entre deux et quatre développeurs en fonction des périodes. L'équipe a comporté à tour de rôle Henri La, Julien Pession, Anja Bekkelien, Loïc Poisot, Yves Grasset et moi-même. Chacun a travaillé sur un ou plusieurs projets. Bien que les projets ne soient pas toujours dépendants les uns des autres, le domaine de compétence de chaque projet est fortement liés aux autres projets. Les compétences de chacun des membres de l'équipe peuvent donc grandement servir et être réutilisées par les autres membres.

Le *Product Owner*. Son rôle est de représenter la voix des utilisateurs finaux de l'application. Pour mon projet en particulier, ce rôle est tenu par Michel Dériaz.

7.1.2 Le *SCRUM* détaillé

Le *SCRUM* est une méthode de gestion de projet particulièrement adaptée pour les projets informatiques. Son premier point caractéristique est l'agilité. Soit la réévaluation constante de l'orientation du projet. Ceci permet de s'adapter au mieux aux spécifications du mandant. En effet, du côté du mandant il est fréquent que les spécifications soient simplement changées en cours du projet. Du côté de l'exécutant, il est possible que certaines fonctionnalités ne soient pas applicables pour des raisons techniques. Un point d'incompréhension contextuel est fréquent, autrement dit que les spécifications du mandant sont mal comprises par les exécutants. Dans chacune de ces éventualités, il est très avantageux de se rendre compte le plus tôt possible de l'incompréhension afin de redresser le tir pour ne pas avoir fait du chemin en vain.

Son deuxième point caractéristique est l'incrémentation itérative. Soit le fait de fournir au mandant, en un temps fixé à deux semaines de production (le *sprint*), une partie utilisable d'un produit en cours de développement. A chacun des *sprints*, le produit en développement se rapproche de plus en plus du produit final. Ceci permet de donner un maximum de valeur rapidement au produit en développement puisque ce sont les parties

qui ont plus de valeur pour le produit final qui seront mises en place. Comme exemple illustratif, il y a la voiture construite à la main par un artisan qui fabriquera et assemblera la structure (châssis et roues). Puis fabriquera le moteur pour l'associer à la structure afin de permettre le déplacement du véhicule. Après ces structures indispensables, il développera les structures secondaires qui seront utiles à la survie du pilote. Soit freins, direction, habitacle. Enfin, il se focalisera sur les accessoires (non indispensables) de manière à améliorer le confort du pilote. Comme vitres, ventilation, autoradio, etc.

Le *Sprint* est l'unité de temps de base. La durée des *Sprints* peut varier entre une et quatre semaines, mais dans notre équipe de développement, nous avons toujours fait durer les *Sprint* deux semaines. Il est plus facile de ne pas faire varier cette durée, simplement pour favoriser l'organisation. Il est favorable d'avoir des habitudes stabilisées au niveau du temps passé à l'organisation d'une période de *SCRUM.*

C'est durant cette période de temps que sont effectués les développements, ainsi qu'à la fin de cette période que sont présentés les résultats aux *Product Owner*.

La manière de procéder est très structurée. Les spécifications générales sont décidées en début de projet et sont intégrées dans ce qui est appelé le *Product Backlog*. Il faut imaginer cet objet comme une liste de tâches à réaliser ou les plus importantes tâches sont placées en haut de liste. Avant chaque nouveau *Sprint*, il faut prendre un nombre de tâches, pouvant être réalisées dans une période de deux semaines, en haut de liste pour les déplacer dans les tâches à réaliser du *Sprint Backlog*. Si une tâche ne peut pas être réalisée en deux semaines, il faut s'arranger pour la diviser en plusieurs sous-tâches, de manière à pouvoir garder des échéances de deux semaines au maximum. Si une tâche n'a pas pu être terminée dans le temps imparti, celle-ci est automatiquement reportée au *Sprint Backlog* du *Sprint* suivant.

Sur l'ensemble du projet il y a le début de projet qui consiste en l'étape d'insertion des spécifications dans le *Product Backlog*, ensuite de quoi il y a les différents *Sprints* qui se suivent. Afin de mieux comprendre le fonctionnement de la méthode, une représentation des différents artefacts participant à cette méthode de projet est utile (Figure [3] : matrice de préférence, cookies et version).

En cours de *Sprints* il y a une réunion de quinze minutes tous les matins permettant de parler des tâches effectuées et des tâches à réaliser dans la journée en cours. Une réunion avec le *Product Owner* en fin de *Sprint* pour lui présenter ce qui a été produit. Nous avons aussi utilisé cette réunion pour effectuer le décompte de temps nécessaire pour la réalisation des tâches à effectuer lors du *Sprint* suivant, bien que dans une gestion de *SCRUM* standardisée cette évaluation est réalisée pendant l'étape de début de projet. Cette différenciation est utile si les spécifications arrivent en cours de route de projet, comme il est normal que ce soit le cas dans une équipe de recherche. Si nous avions été

une équipe de développement de logiciel, il aurait été logique de suivre le Scrum plus précisément.

7.2 FoxyTag

7.2.1 Utilité

Foxytag.com est un site qui sert à faire la promotion du moteur de confiance entre acteurs d'un réseau. Sa vocation est d'utiliser ce moteur de confiance pour valoriser et donner du poids à des annonces de radars, dans des applications mobiles d'*avertisseur de radars*. Pour ceci, des applications mobiles ont été développées. Sur ces applications, plus les personnes signalent un lieu de contrôle de vitesse, plus ils vont tisser des liens de confiance avec les personnes qui auront déjà signalé le même lieu. Ainsi, ces applications pourront signaler des zones critiques aux personnes de confiance en leur annonçant arriver proche d'un radar, ceci avant de se trouver sur la zone. Afin d'utiliser ce moteur de confiance, il faut pouvoir s'identifier de manière à correspondre à un profil unique et ainsi accroître de manière significative les liens de confiance avec d'autres personnes. Plus les liens de confiance seront développés, plus les annonces seront pertinentes.

Le site propose en plus de quelques applications spécifiques de localisation de radars, d'un système d'enregistrement. Ce système est aussi accessible depuis toutes les applications qui utilisent le moteur de confiance.

7.2.2 Pertinence des données

Le *username*, aussi appelé login, est l'identifiant de l'utilisateur. Il est personnel et unique. C'est grâce à cet attribut qu'un utilisateur pourra s'identifier ou qu'il pourra être identifié par l'administration du système.

Le *password*, ou mot de passe, est pour le site FoxyTag une chaîne de caractères. Quand l'utilisateur va insérer ses données, lors de son inscription, il va associer à son nom une chaîne de caractères, le *password*. C'est cette chaîne de caractères codée grâce à l'algorithme de hachage SHA-1 qui va être stockée dans les attributs associés à l'utilisateur. A chaque fois que l'utilisateur voudra se connecter, il rentrera son *username* et sont *password*. Les deux attributs seront envoyés au système qui vérifiera si le password (de nouveau haché avec le même algorithme) correspond à la chaîne de caractère du password associée au *username* envoyé. Si un tel mécanisme est sûr, c'est parce que le système garanti qu'il n'y ait pas deux utilisateurs différents qui utilisent le même identifiant pour leur *username*, et parce qu'il n'est pas (sans des moyens lourds) possible de retrouver le mot de passe de l'utilisateur en possédant la chaîne de caractère hachée du mot de passe. En effet, les algorithmes de hachage sont irréversibles, car ils

répondent à la particularité mathématique du défaut d'*injectivité* (Figure [31] : Non injectivité).

Le *phone*, ou numéro de téléphone de l'utilisateur, est un numéro unique. Pour les validations à faire lors de l'inscription ou de changement de données, c'est l'outil que l'utilisateur va faire servir pour communiquer avec le système. Lors de l'enregistrement des données, le système vérifie que celui-ci ne soit pas déjà utilisé dans la base de données à l'inscription de l'utilisateur, de manière à être sûr qu'un utilisateur avec un seul numéro de téléphone puisse posséder différents *username*. Si c'était le cas, il serait possible qu'un utilisateur puisse déstabiliser le moteur de confiance en adoptant des comportements différents. Par exemple, en utilisant deux comptes différents, il pourrait valider une annonce qu'il aurait faite lui-même. Il pourrait aussi surcharger le système sans avoir de moyens industriels à disposition en utilisant des nouveaux réseaux de confiance qu'il aurait créé. Pour ces raisons, il est important que chacun des utilisateurs ait la même importance que les autres dans le moteur de confiance.

Le *mail* est l'adresse personnelle que le système va utiliser pour communiquer avec l'utilisateur. Il serait possible de faire les communications avec l'utilisateur par SMS, grâce au numéro unique *phone* de chaque utilisateur, mais pour une raison de coûts, le choix s'est porté sur le email, qui par extension des contraintes de *phone*, doit aussi être unique. Il y a une asymétrie de moyen de communication. En effet, l'utilisateur doit communiquer avec le système en utilisant le SMS, alors que le système communique avec l'utilisateur en utilisant le mail.

7.2.3 Les données

Les données des différents utilisateurs sont stockées sur un serveur. Afin de fournir ces données, un utilisateur se rend sur le site de foxytag et rentre ses données personnelles. L'outil développé durant ce travail sert à fournir une validation des données de la personne qui s'inscrit non plus en passant par le serveur, mais de faire la démarche de validation de ces données depuis son propre ordinateur ou téléphone mobile. Autrement dit, en local. L'outil préexistant permettait d'envoyer les données de l'utilisateur au travers d'une requête GET au serveur. Si les données étaient valides, le serveur insérait les données dans une table temporaire et demandait, par un message dans une nouvelle page HTML, d'effectuer une confirmation grâce au mail envoyé à l'adresse mail inséré par l'utilisateur (Figure [32] : outil SmsRegistration cmd=register). Si les données n'étaient pas bonnes, le serveur n'insérait pas les données et indiquaient par un message dans une nouvelle page HTML que les données étaient invalides. Que ce soit parce que les données insérées existaient déjà (Figure [33] : outil SmsRegistration cmd=register données déjà utilisées) ou que ce soit parce que le format utilisé pour les données n'était pas valide(Figure [34] : outil SmsRegistration cmd=register données invalides).

7.3 Différents langages

7.3.1 Description du HTML

Le HTML5 est utile à la mise en forme du contenu de pages web. Dans le cadre de ce travail, l'intérêt de la version 5 par rapport à la version précédente est la présence d'outils utiles. Dans une balise <input> utile pour l'insertion de données lors de la création de formulaires, de nouveaux attributs font particulièrement l'affaire. Le type "email" est utile à l'insertion de l'email qu'utilisera l'utilisateur pour se faire confirmer son enregistrement. Le nouveau type "number" pourrait être utile à l'insertion du numéro de téléphone de l'utilisateur, mais ce type ne nous permet pas d'être sûrs qu'il utilise un numéro de téléphone correspondant au format international. Pour ceci, il y a le nouvel attribut "pattern" qui permet d'insérer une expression rationnelle permettant de vérifier le contenu de la balise input. Si le contenu correspond à l'expression rationnelle le formulaire pourra être envoyé, alors que si ce n'est pas le cas, une alerte sera indiquée sur le champ invalide. C'est sans aucun doute le plus utile des attributs, car on peut définir très précisément ce qui est permit comme chaîne de caractères.

7.3.2 Description du JavaScript

Le JavaScript permet une fonctionnalité particulière que le HTML5 ne permet pas. Il permet à la page présentée à l'utilisateur d'adopter certains états en fonction du comportement de l'utilisateur. En l'occurrence, il est utile ici de pouvoir réaliser la validation des expressions rationnelles en cours d'insertion, sans attendre que l'utilisateur fasse l'action d'envoi du formulaire. Ceci est donc une raison particulière pour faire réaliser le travail de validation au langage JavaScript, et non au HTML5. L'utilisateur comprend en même temps qu'il insère les données si celles-ci seront acceptées ou non.

7.3.3 Description de jQuery

jQuery a été utilisé pour le moteur de validation ainsi que pour les requêtes Ajax effectuées à un serveur externe par requêtes GET. Il étant et simplifie les commandes communes à javaScript[30].

7.3.4 Description des expressions rationnelles

Une expression rationnelle est une chaîne de caractères C_1 (appelée regex). Un moteur d'expressions rationnelles permet de définir si la regex est contenue dans une chaîne de caractères C_0. Il existe différentes manières de vérifier. La méthode DFA qui est la plus

[30] Source : http://fr.wikipedia.org/wiki/JQuery

rapide. Elle ne va vérifier chacun des caractères du string examiné qu'une seule fois au plus.

La méthode NFA est un peu plus développée. Deux types de moteurs la composent. Les moteurs NFA traditionnels, ainsi que les moteurs NFA POSIX. Les premiers vont comparer chaque élément du C_1 au C_0, alors que les seconds moteurs vont en plus récupérer toujours l'élément le plus long et le plus à gauche dans la chaîne C_0.

7.3.5 Description du *Css*

Le CSS est utile à la présentation des documents HTML. Pour faciliter la compréhension de l'usage du CSS, l'analogie avec le design pattern "Model-Vue-Contrôleur" est pratique. L'élément "Model" serait équivalent au code fourni par le langage HTML. Le code fourni par le CSS joue le rôle de "Vue".

7.3.6 Description du *PHP*

L'exemple du "MVC" peut être étendu pour dire que le code écrit dans le langage PHP joue le rôle du contrôleur. Il faut mettre le doigt sur le fait que le langage PHP est, contrairement aux autres langages, non pas exécuté du côté client, mais exécuté du coté serveur. Il est donc particulièrement puissant puisqu'il peut présenter des contenus différents en fonction du client qui effectue la requête.

7.3.7 Description d'*Ajax*

Le langage Ajax sert principalement à effectuer des requêtes depuis un client sur un serveur, sans pour autant qu'il y ait un changement de page. En fonction du comportement détecté par le langage JavaScript, la page elle-même peut faire des requêtes de contenus à un serveur.

7.4 Prototypes et outils rendus

7.4.1 Prototype *PHP*

Traitant les différentes données d'un utilisateur grâce aux pages de contrôle[31] (Figure [21] : structure des pages du site, avec pages de contrôle).

[31] Cf. Annexe DVD : Prototype PHP

7.4.2 Prototype *Ajax*

Traitant les différentes données d'un utilisateur grâce aux fonctions javaScript et Ajax[32] (Figure [22] : structure des pages du site, avec requêtes *Ajax*).

7.4.3 Version allégée pour téléphones mobiles

Traitant les différentes données d'un utilisateur grâce aux fonctions *javaScript* côté client, au *PHP* côté serveur et à l'appel récursif des pages visitées par l'utilisateur[33]. Son intérêt est de ne pas utiliser de grosses librairies comme *jQuery* ou de moteur de validation qui bien qu'ils favorisent grandement la présentation et la dynamique d'un site, alourdissent passablement le code. Ceci provoque de longs délais de chargement et des surcoûts pour les utilisateurs de téléphones portables. Pour cette version particulière, le mandant a fait cinq demandes précises.

7.4.3.1 Validation

Exigence du mandant :

La validation doit être effectuée en affichant un texte d'erreur et un bord rouge des champs étant remplis. Le bord rouge ne doit apparaître la première fois que lorsque l'utilisateur quitte un champ invalide.

Réalisation :

Afin de réaliser la validation des champs, des fonctions *JavaScript* sont appelées lorsque des évènements sont décelés sur les champs présentés par le browser. Ces évènements sont onBlur, onChange et onKeyUp. Le premier s'active quand le champ est quitté, il va provoquer la mise du bord du champ en vert s'il est valide et en rouge s'il ne l'est pas. Il indique aussi que le champ quitté pour la première fois a été visité. Lorsque qu'il sera visité les fois suivantes, il indiquera automatiquement les textes d'erreurs s'il ne correspond pas au format attendu.

Les deux évènements suivants provoquent le contrôle de la validation du champ en question à chaque fois que le texte est modifié par la saisie automatique ou qu'une touche est relâchée.

---

[32] Cf. Annexe DVD : Prototype Ajax

[33] Cf. Annexe DVD : Version allégée

7.4.3.2 Bouton d'envoi

Exigence du mandant :

Le bouton d'envoi des formulaires ne doit s'activer que lorsque l'ensemble des champs sont valides.

Réalisation :

La fonction de contrôle de la validité de l'ensemble des champs est appelée à deux moments particuliers. Le premier l'est au chargement de la page et le second se fait à chaque fois qu'un champ est contrôlé comme valide. Cette fonction « validateAllData » permet de mettre le bouton d'envoi du formulaire à sa forme active si tous les chaînes de caractères sont contrôlées comme respectant leurs expressions régulières respectives.

7.4.3.3 Récupération des messages de retour

Exigence du mandant :

Les messages retournés par le serveur doivent être correctement appréhendés

Réalisation :

Lors de l'envoi des données au serveur oxgva, chacune des pages va réagir en fonction du formulaires qu'elles contiennent. Elles vont s'attendre à des résultats particuliers. La variable de contrôle de ses résultats indique le comportement à adopter. Elle peut indiquer qu'elle n'a pas été initialisée et ainsi montrer que la page est demandée pour la première fois. Si elle indique que le retour est valide, le traitement *javaScript* va cacher le formulaire et donner des instructions à suivre. Si la variable de contrôle indique un retour invalide, alors la page va afficher le message de retour du serveur de manière à indiquer à l'utilisateur les problèmes à corriger pour véritablement valider ses actions

7.4.3.4 Tooltip

Exigence du mandant :

Après chaque champ, une petite icone sous forme de point d'interrogation doit être affichée. Cette icone doit permettre en passant la souris par-dessus d'indiquer les règles de validation.

Réalisation :

Pour réaliser ceci, une image sous forme de point d'interrogation a été ajoutée à côté des champs des formulaires d'enregistrement et de modification des données de manière à ce que l'utilisateur puisse comprendre les différentes types d'informations qu'on lui

demande. Il suffit d'insérer une variable de titre avec le texte de l'information dans la balise de l'image. Les propriétés des browsers modernes permettent l'interactivité de l'utilisateur avec les images respectant le comportement attendu par le mandant.

7.4.3.5 Contenus permanant et dynamiques

Les quatre pages doivent contenir aussi bien le contenu générique que dynamique

Réalisation :

Il est particulièrement judicieux d'utiliser le langage PHP afin de bénéficier de sa dynamique avec son traitement sur le serveur. Par contre une fois que la page aura été affichée par le browser, il n'y a que la dynamique javaScript qui pourra être utilisée pour les champs de validation, comme décrit plus haut. Une fausse dynamique browser sera mise en place lors des envois des formulaires. L'utilisateur va avoir l'impression que la page n'a pas changé, alors qu'à la soumission du formulaire il va provoquer un appel récursif de la page sur laquelle il se trouve. Il va donc provoquer une transmission de données au serveur et à un traitement de celles-ci avant le réaffichage de la page avec un contenu différent.

Son premier avantage est l'utilisation de variables stockées sous forme de tableau associatif. Elles permettent ainsi de reconnaître la langue courante demandée par l'utilisateur en recevant une variable lorsque celui-ci clique sur un drapeau. La variable connaît aussi son nom, donc sa responsabilité. Il est alors possible pour le serveur d'aller récupérer dans le dictionnaire l'association entre le nom de la variable et la langue dans laquelle elle doit être affichée et de la fournir comme contenu de la page à afficher.

Le deuxième traitement dynamique est les réponses aux requêtes faites au serveur oxgva. Ses retours sont traités sur le serveur *foxytag* qui va attribuer une valeur à une variable de contrôle. Lorsque le browser recevra la page à afficher, une variable *javaScript* sera appelée à son chargement. Le code *javaScript* permettant des comportements différents en fonction de certaines données, il va permettre l'affichage ou le masquage des formulaires ainsi que des textes d'alerte grâce à la variable acquise coté serveur.

7.4.4 Moteur de validation

Permettant la validation à la volée et utilisant jQuery[34].

[34] Cf. Annexe DVD : Moteur validation

7.4.5 Wrapper *PHP*

Permettant la transmission des requêtes du browser au serveur interrogé en utilisant un serveur intermédiaire pour hoster le wrapper et détourner les règles standard de sécurité[35].

7.5 Guide utilisateur

7.5.1 L'installation

Afin d'utiliser un des trois prototypes développés pour ce travail, il suffit d'insérer l'un des dossiers fournis[36] sur un serveur de type Wamp Serveur ou sur un serveur de production.

Les requêtes Ajax sont faites au serveur oxgva mis à disposition par le mandant. Pour ce qui est des requêtes SQL utilisées pour le prototype PHP, les droits ne sont accessibles que depuis le domaine de l'université. Pour une question de temps, je n'ai pas pu développer une base de données locale.

En cas de volonté d'effectuer une insertion des outils utilisés à une adresse différente, pour leur contrôle absolu, il faudra modifier le code des pages PHP de manière à effectuer des appels à leurs adresses respectives. Pour cette raison, les points suivants traitent des adresses relatives des différents outils dans les prototypes rendus.

7.5.1.1 jQuery

Le code de jQuery se trouve dans le dossier js.

7.5.1.2 Moteur de validation

Le moteur de validation est stocké dans le sous-dossier js/jsValidation pour le code javaScript et dans css/cssValidation pour le code des feuilles de style.

7.5.1.3 Wrapper Php

Les wrappers sont intégrés dans le sous-dossier php/AjaxCrossWrapper.php pour être utilisés.

---

[35] Cf. Annexe DVD : Wrapper PHP

[36] site_v06_fonctionnelPhpLight, site_v06_fonctionnelPhp, site_v08_fonctionnelAjaxLight ou site_v08_fonctionnelAjax

7.5.1.4 Version allégée

Afin de bénéficier au maximum de la réutilisabilité, les codes java et php ont été centralisés. Chacune des pages les utilisant s'y référant. Il y a ainsi l'évitement de la redondance (Figure [35] : Structure centralisée). Afin de comprendre l'avantage de cette structure, la visualisation des mécanismes globaux des fonctionnements des structures et architectures d'un serveur peut être utile (Figure [36] : Architecture Serveur – Browser).

7.5.2 Utilisation

7.5.2.1 Enregistrement

Afin d'enregistrer un utilisateur il faut se diriger sur la page de login[37] et cliquer sur le bouton Create an account (Figure [37] : Page de login). Sur la page d'enregistrement (Figure [38] : Page d'enregistrement) d'un utilisateur, il faut entrer cinq chaînes de caractères. Attention, chacun des champs de caractère respecte un standard d'insertion visible dès que l'on commence à modifier les champs et du temps qu'il n'est pas valide. Lors de l'envoi du formulaire, il est possible que les données déjà entrées soient déjà utilisées. Dans ce cas, une page contenant une alerte s'affiche (Figure [39] : Alerte de données déjà utilisées). Lorsqu'un enregistrement est réussi, la page (Figure [40] : Confirmation d'enregistrement) est affichée et il faudra effectuer une confirmation par SMS[38]. Les informations relatives à l'enregistrement sont envoyées par mail (Figure [41] : Demande de confirmation par mail et Figure [42] : Confirmation par mail).

7.5.2.2 Login

Afin qu'un utilisateur retrouve ses données, il peut accéder par la page de login en insérant son username et son mot de passe. Il faut bien faire attention de ne pas entrer de mauvaises valeurs qui seraient cause d'alertes de validation ou d'une annonce d'insertion invalide (Figure [43] : Annonce de mauvais). Ensuite, il va cliquer sur le bouton login (Figure [44] : Clic pour se logger), pour accéder à la page d'un utilisateur.

7.5.2.3 Modification des données

Depuis la page de l'utilisateur (Figure [45] : Page d'un utilisateur) d'accéder à une page de modification des données utilisateurs (Figure [46] : Modification des données). Il est

---

[37] site_v06_fonctionnelPhp/en/loginPage.php ou site_v08_fonctionnelAjax/en/loginPage.php

[38] Il est possible d'utiliser la commande « http://www.oxgva.com/SmsRegistrationVpa/SmsRegistration?NUMBER=0041781234567&MESSAGE=FoxyTag%20reg%20xxx ». Attention de bien modifier le numéro de téléphone ainsi que les trois derniers caractères (reçu par mail).

possible d'uniquement changer sa langue et ainsi directement recevoir une confirmation (Figure [47] : Confirmation de modification de la langue).

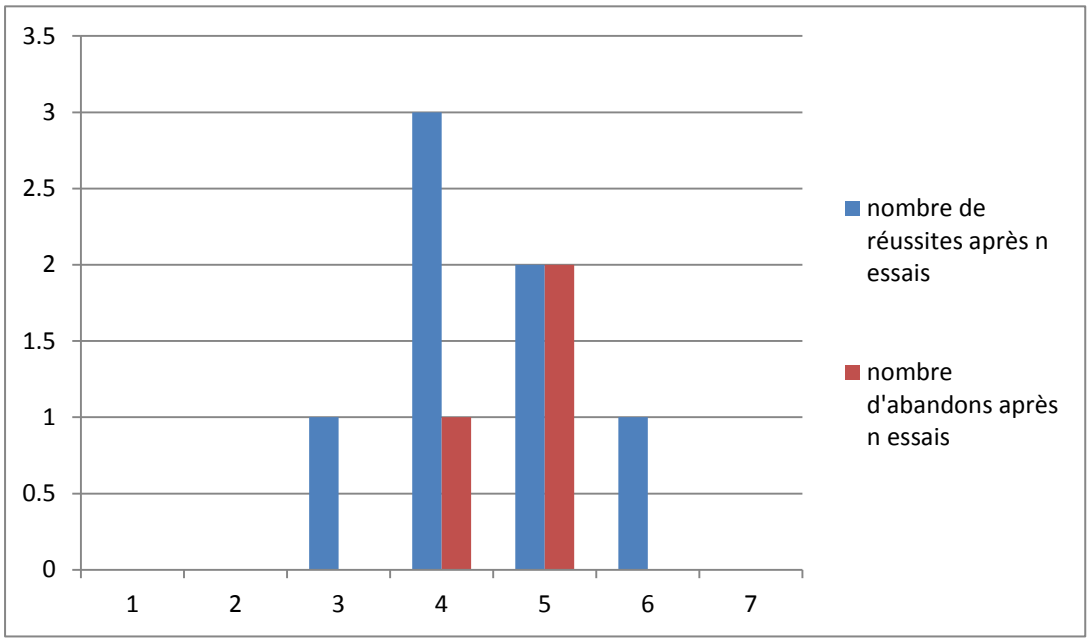
Si l'utilisateur veut modifier d'autres données, il doit de nouveau accéder à la page de modification de ses données (Figure [48] : Page de modification des données) et changer les champs correspondants. Une page de confirmation lui est envoyée (Figure [49] : Confirmation). Il y a alors les envois de mail (Figure [50] : Demande de confirmation par mail), comme pour l'enregistrement servant à la confirmation finale des données envoyées (Figure [51] : Confirmation par mail).

7.5.2.4 Récupération de mot de passe

Depuis la page de login (Figure [52] : Page de login) il est possible d'accéder à une page de récupération de son mot de passe (Figure [53] : Page de récupération de mot de passe). Depuis là, un utilisateur entrant une adresse mail n'étant pas listée dans la base de donnée serait averti par un texte d'alerte (Figure [54] : Alerte d'adresse mail inexistante). S'il entre une bonne adresse mail, il reçoit une confirmation (Figure [55] : Confirmation d'envoi de mail) d'attribution d'un nouveau mot de passe qu'il peut récupérer dans son courrier (Figure [56] : Réception du nouveau mot de passe).

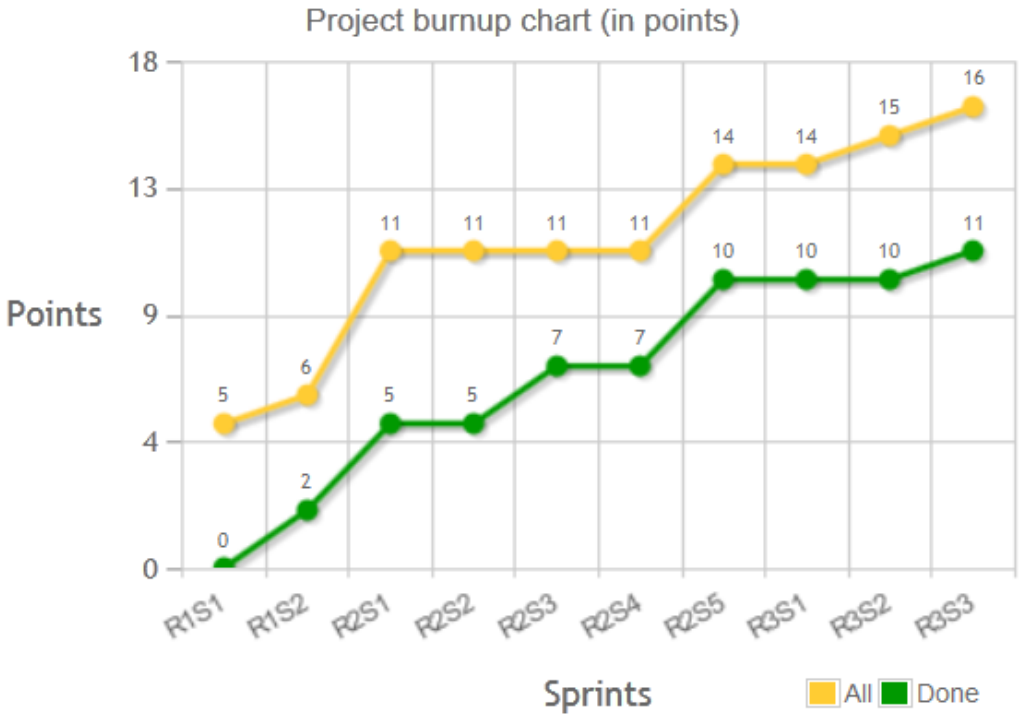
8. Liste des Figures

Figure [1] : Tentatives pour insertion de données valides



(Source: Marc Falcy)

Figure [2] : Burnup chart final



(Source : Marc Falcy)

Figure [3] : matrice de préférence, cookies et version

Critères :

A	Utilisation généralisée sur les browsers
B	Sécurité
C	Récolte d'information sur le comportement de l'utilisateur
D	Temps de formation à l'outil
E	Facilité de mise en œuvre de l'outil

Matrice de préférence :

	A	B	C	D	E
A					
B	A				
C	A	C			
D	A	D	D		
E	A	E	E	E	

Résultats :

Critères	Cookies	Représentation	Session	Représentation
A	non	0	oui	4
B	oui	0	oui	0
C	oui	1	non	0
D	non	0	oui	2
E	oui	3	oui	3

Total :

Cookies : 40%

Session : 90%

(Source : Marc Falcy)

Figure [4] : Matrice de préférence, validation *javaScript*

Critères :

A	Temps de réalisation
B	Temps d'apprentissage
C	Qualité de la documentation
D	Présence d'exemples et d'une communauté sur le Net

Matrice de préférence :

	A	B	C	D
A				
B	B			
C	C	C		
D	D	D	C	

Résultats :

Choix 1 : Réalisation personnelle des fonctions *javaScript*

Choix 2 : Utilisation du plugin jQuery de Cédric Dugas (www.position-relative.net)

Choix 3 : Utilisation du moteur de validation d'Alec Hill (www.livevalidation.com)

Critères	Choix 1	Représentation	Choix 2	Représentation	Choix 3	Représentation
A	non	0	oui	0	oui	0
B	non	0	non	0	oui	1
C	non	0	oui	3	oui	3
D	non	0	oui	2	non	0

Total :

Choix 1: 0%

Choix 2: 83%

Choix 3: 66%

(Source : Marc Falcy)

Figure [5] : Appel du moteur de validation

```
<!-- textForm login par Marc Falcy-->
 <tr>
    <td width="130" class="textForm"> </td>
    <td width="180"  class="textForm" for="username" >Login</td>
    <td width="250"> <input class="validate[required,custom[loginName],custom[loginSpace]]
                        text-input" type="text" name="login" value="" id="username"> </td>
 </tr>

 <!-- textForm password par Marc Falcy-->
 <tr>
    <td width="130" class="textForm"> </td>
    <td width="180"  class="textForm" for="password" >Password</td>
    <td width="250"> <input class="validate[required,length[7,25]]
                        text-input" type="password" name="password" value="" id="password"> </td>
 </tr>
```

(Source : Marc Falcy)

Figure [6] : Modification des règles validantes

```
   "telephone":{
        "regex":"/^([+]{1}[0-9]{8,16})$/",
        "regex":"/^+|(00)[0-9]{2}[0-9]{9}$/",
```

(Source : Marc Falcy)

Figure [7] : Différents comportements de l'utilisateur

```
 if(settings.inlineValidation == true){      // Validating Inline ?

    //$(this).not("[type=checkbox]").bind("blur", function(caller){loadValidation(this)})
    $(this).not("[type=checkbox]").bind("keyup", function(caller){loadValidation(this)})
    //$(this).not("[type=checkbox]").bind("keydown", function(caller){loadValidation(this)})
    $(this).not("[type=checkbox]").bind("change", function(caller){loadValidation(this)})
    $(this).not("[type=checkbox]").bind("onFocus", function(caller){loadValidation(this)})
    $(this+"[type=checkbox]").bind("click", function(caller){loadValidation(this)})
 }
```

(Source : Marc Falcy)

Figure [8] : Page d'enregistrement

(Source : Nouvelle version d'enregistrement sur foxyTag. Marc Falcy)

Figure [9] : Expression rationnelle de required

```
allRules = {"required":{
                    "regex":"none",
                    "alertText":"* This field is required",
                    "alertTextCheckboxMultiple":"* Please select an option",
                    "alertTextCheckboxe":"* This checkbox is required"},
                "length":{
                    "regex":"none",
                    "alertText":"*Between ",
                    "alertText2":" and ",
                    "alertText3": " characters allowed"},
```

(Source : moteur de validation, https://github.com/posabsolute/jQuery-Validation-Engine)

Figure [10] : Expression rationnelle étendue

```
/* VALIDATION FUNCTIONS */
function _required(caller,rules){   // VALIDATE BLANK FIELD
    callerType = $(caller).attr("type")

    if (callerType == "text" || callerType == "password" || callerType == "textarea"){

        if(!$(caller).val()){
            isError = true
            promptText += settings.allrules[rules[i]].alertText+"<br />"
        }
    }
    if (callerType == "radio" || callerType == "checkbox" ){
    if (callerType == "select-one") { // added by paul@kinetek.net for select boxes, Thank you
    if (callerType == "select-multiple") { // added by paul@kinetek.net for select boxes, Thank you
}
```

(Source : moteur de validation, https://github.com/posabsolute/jQuery-Validation-Engine)

Figure [11] : Expression rationnelle de loginName

```
    "loginName":{
        "regex":"/^[ a-zA-Z0-9ÀÁÂÃÄÅÆàáâãäåæÇçÐðÈÉÊËèéêëÌÍÎÏìíîïÑñÒÓÔÕÖØŒòóôõöøœŠšÙÚÛÜùúûüÝŸýÿŽž]{3,25}$/",
        "alertText":"* Invalid login, enter from 3 to 25 accepted symbols"},
```

(Source : https://github.com/posabsolute/jQuery-Validation-Engine.

Modifiée par Marc Falcy)

Figure [12] : Expression rationnelle des loginSpace

```
    "loginSpaceBE":{
        "regex":"/^[^ ].*[^ ]$/",
        "alertText":"* No spaces at the begining or the end"},
    "loginSpace":{
        "regex":"/^[^ ]+( [^ ]+)*$/",
        "alertText":"* No more than one consecutive space and no space at the begining or the end"},
```

(Source : https://github.com/posabsolute/jQuery-Validation-Engine.

Modifiée par Marc Falcy)

Figure [13] : Expression rationnelle de *mail*

```
"email":{
    //"regex":"/^[a-zA-Z0-9_.-]+@([a-zA-Z0-9_.-]{2,}\.)[a-z]{2,4}$/",
    "regex":"/[\\w%-]+(\\.[\\w%-]+)*@[\\w%-]+(\\.[\\w%-]+)*\\.[\\w%-]{2,4}/",
    "alertText":"* Invalid email address"},
```

(Source : https://github.com/posabsolute/jQuery-Validation-Engine.

Modifiée par Marc Falcy)

Figure [14] : Expression rationnelle *phone*

```
"telephone":{
    "regex":"/^([+]{1}[0-9]{8,16})$/",
    "regex":"/^+|(00)[0-9]{2}[0-9]{9}$/",
```

(Source : https://github.com/posabsolute/jQuery-Validation-Engine.

Modifiée par Marc Falcy)

Figure [15] : Expression rationnelle de longueur

```
function _length(caller,rules,position){     // VALIDATE LENGTH

    startLength = eval(rules[position+1])
    endLength = eval(rules[position+2])
    feildLength = $(caller).attr('value').length

    if(feildLength<startLength || feildLength>endLength){
        isError = true
        promptText += settings.allrules["length"].alertText+startLength
        +settings.allrules["length"].alertText2+endLength
        +settings.allrules["length"].alertText3+"<br />"
    }
}
```

(Source : https://github.com/posabsolute/jQuery-Validation-Engine)

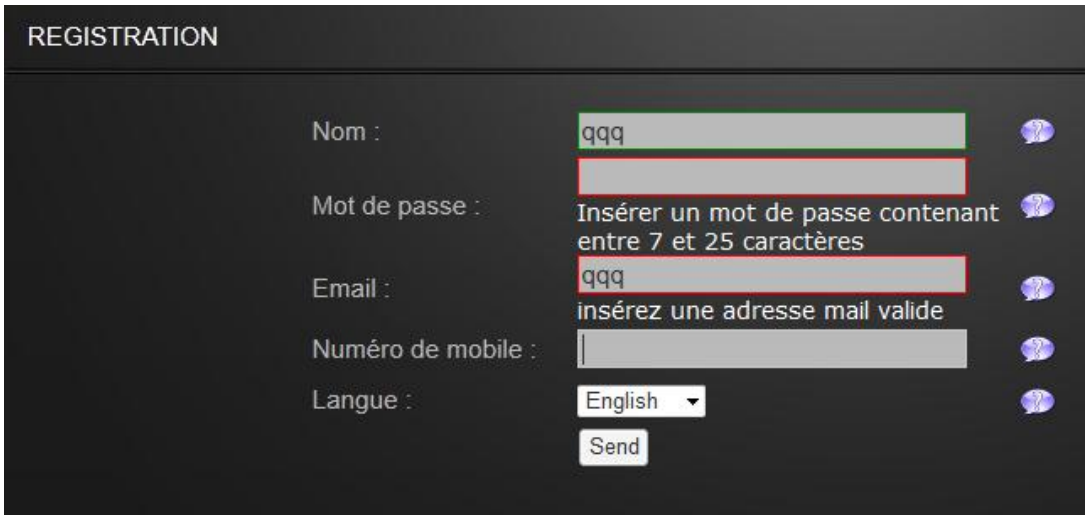
Figure [16] : Expression rationnelle de confirmation de mot de passe

```
    "confirm":{
        "regex":"none",
        "alertText":"* Your field is not matching"},
```

(Source : https://github.com/posabsolute/jQuery-Validation-Engine)

Figure [17] : affichage des messages d'alerte

(Source : Nouvelle version d'enregistrement sur foxyTag. Marc Falcy)



(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

Figure [18] : position et la taille de la boîte d'alerte

```
/* ci-dessous la forme de la box d'alerte d'erreur*/
.formError .formErrorContent {
    /*width:100%; */
    width:300%;
    /*background:#000;*/
    background:#f00;
    color:#fff;
    font-family:tahoma;
    font-size:10px;
    box-shadow: 0px 0px 6px #000;
    -moz-box-shadow: 0px 0px 6px #000;
    -webkit-box-shadow: 0px 0px 6px #000;
    padding:4px 10px 4px 10px;
    border-radius: 6px;
    -moz-border-radius: 6px;
    -webkit-border-radius: 6px;
}
```

(Source : https://github.com/posabsolute/jQuery-Validation-Engine.

Modifiée par Marc Falcy)

Figure [19] : position de la flèche de la boîte d'alerte

```
/* ci-dessous la position de la flèche indiquant
    la textbox où se trouve l'erreur*/
.formError .formErrorArrow{
    position:absolute;
    bottom:0;left:20px;
    width:15px; height:15px;

}
```

(Source : https://github.com/posabsolute/jQuery-Validation-Engine.

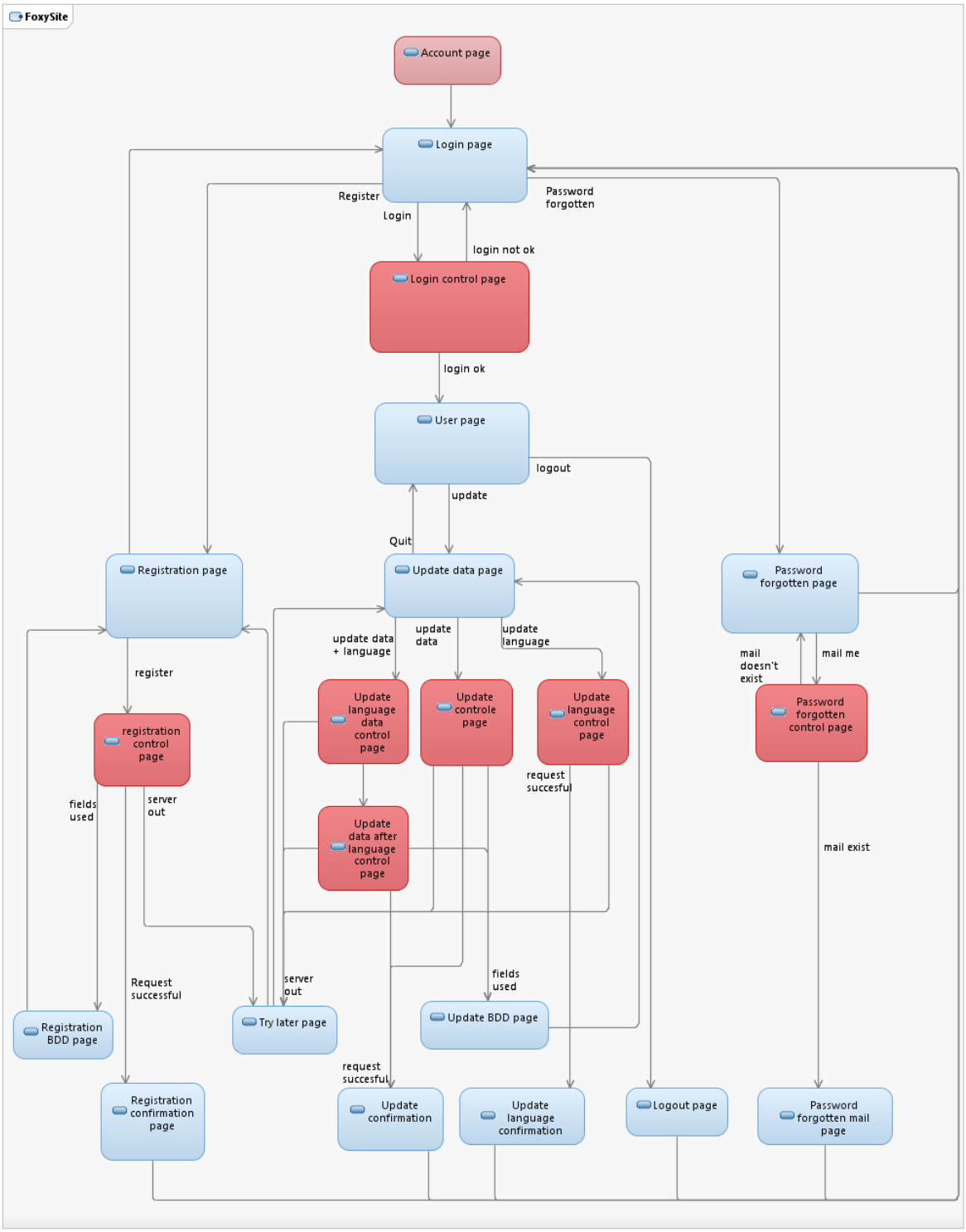
Modifiée par Marc Falcy)

Figure [20] : taille de la flèche de la boîte d'alerte

```
/* ci-dessous la flèche indiquant la textbox où se trouve l'erreur*/
.formError .formErrorArrow .line10
{width:15px;height:1px; background:#f00;margin:0 auto; font-size:0px; display:block;}
.formError .formErrorArrow .line9
{width:13px;height:1px; background:#f00;margin:0 auto;display:block;}
.formError .formErrorArrow .line8
{width:11px;height:1px; background:#f00;margin:0 auto;display:block;}
.formError .formErrorArrow .line7
{width:9px;height:1px; background:#f00;margin:0 auto;display:block;}
.formError .formErrorArrow .line6
{width:7px;height:1px; background:#f00;margin:0 auto;display:block;}
.formError .formErrorArrow .line5
{width:5px;height:1px; background:#f00;margin:0 auto;display:block;}
.formError .formErrorArrow .line4
{width:3px;height:1px; background:#f00;margin:0 auto;display:block;}
.formError .formErrorArrow .line3
{width:1px;height:1px; background:#f00;margin:0 auto;display:block;}
```

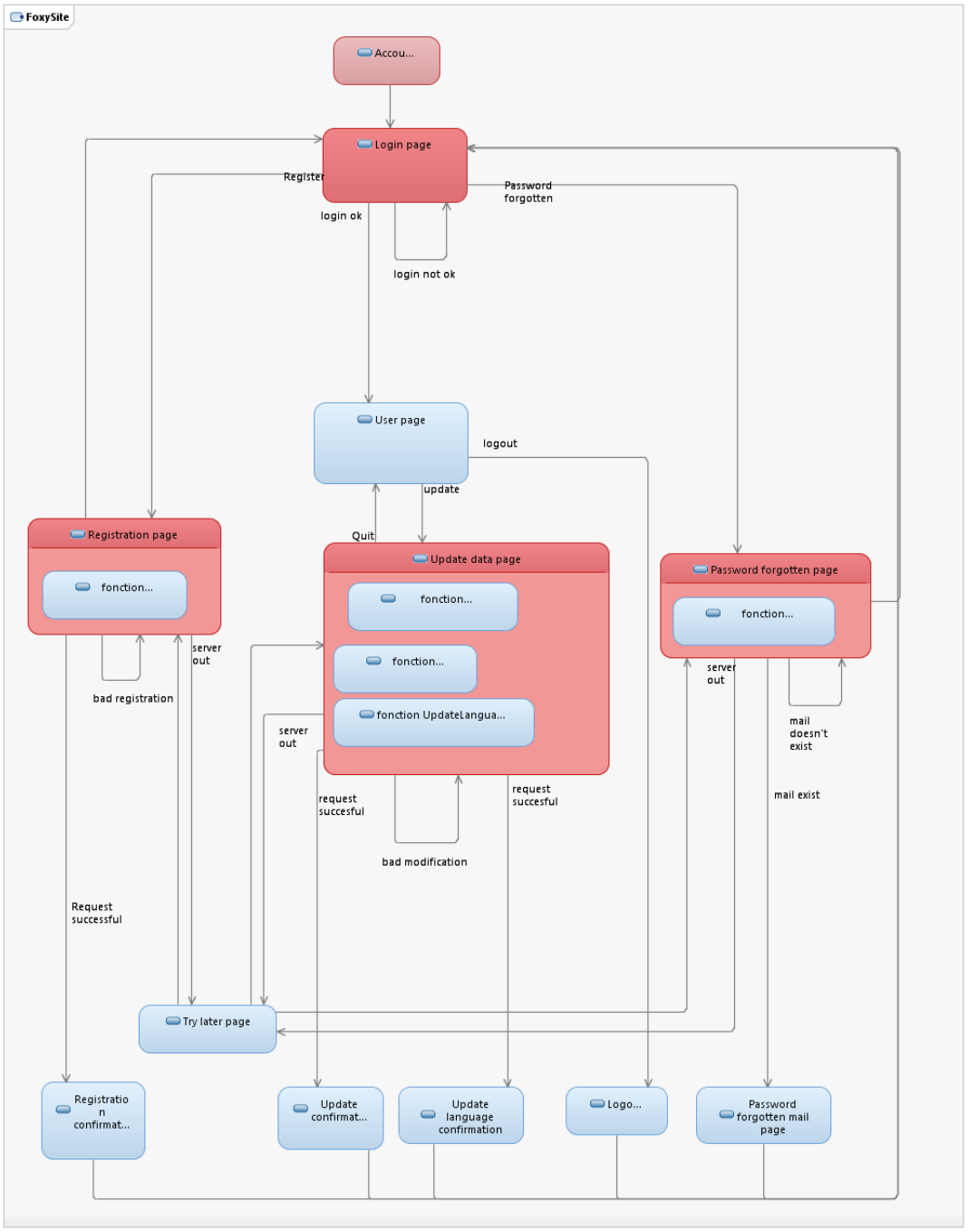
(Source : https://github.com/posabsolute/jQuery-Validation-Engine.

Figure [21] : structure des pages du site, avec pages de contrôle



(Source : Marc Falcy)

Figure [22] : structure des pages du site, avec requêtes *Ajax*



(Source : Marc Falcy)

Figure [23] : requête Ajax. Récupération de mot de passe

```
//recover email
$.ajax({  // var request =
    type: "GET",
    async: false,
    data: {
        mail: email,
    },
    url :"../php/AjaxCrossWrapper.php/AjaxCrossWrapperRecoverMail
    ".php?url=HTTP://www.oxgva.com/SmsRegistrationVpa/SmsRegistration?cmd=recover",
})
.error(function(jqXHR, textStatus, errorThrown){
})
.done(function(data, textStatus, jqXHR){
    xResMail = jqXHR.responseText;
})
;
```

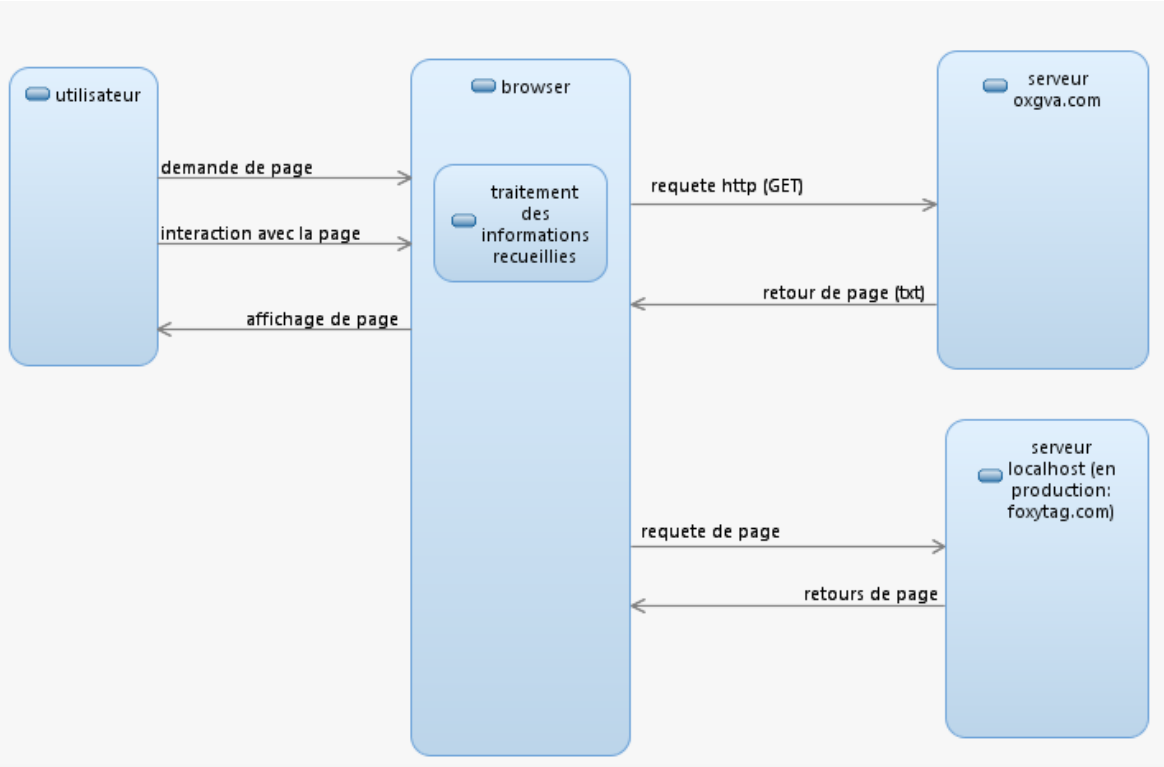
(Source : Marc Falcy)

Figure [24] : requête Ajax. Vérification des données

```
// contrôle possibilité d'insertion
$.ajax({  // var request =
    type: "GET",
    async: false,
    data: {
        username: name,
        mail: mail,
        phone: phone,
        language: language,
        password: pwd,
    },
    url :"../php/AjaxCrossWrapper.php/AjaxCrossWrapperCheckRegisterData.
    "php?url=HTTP://www.oxgva.com/SmsRegistrationVpa/SmsRegistration?cmd=checkData",
})
.error(function(jqXHR, textStatus, errorThrown){
})
.done(function(data, textStatus, jqXHR){
    xResCheck = jqXHR.responseText;
})
;
```

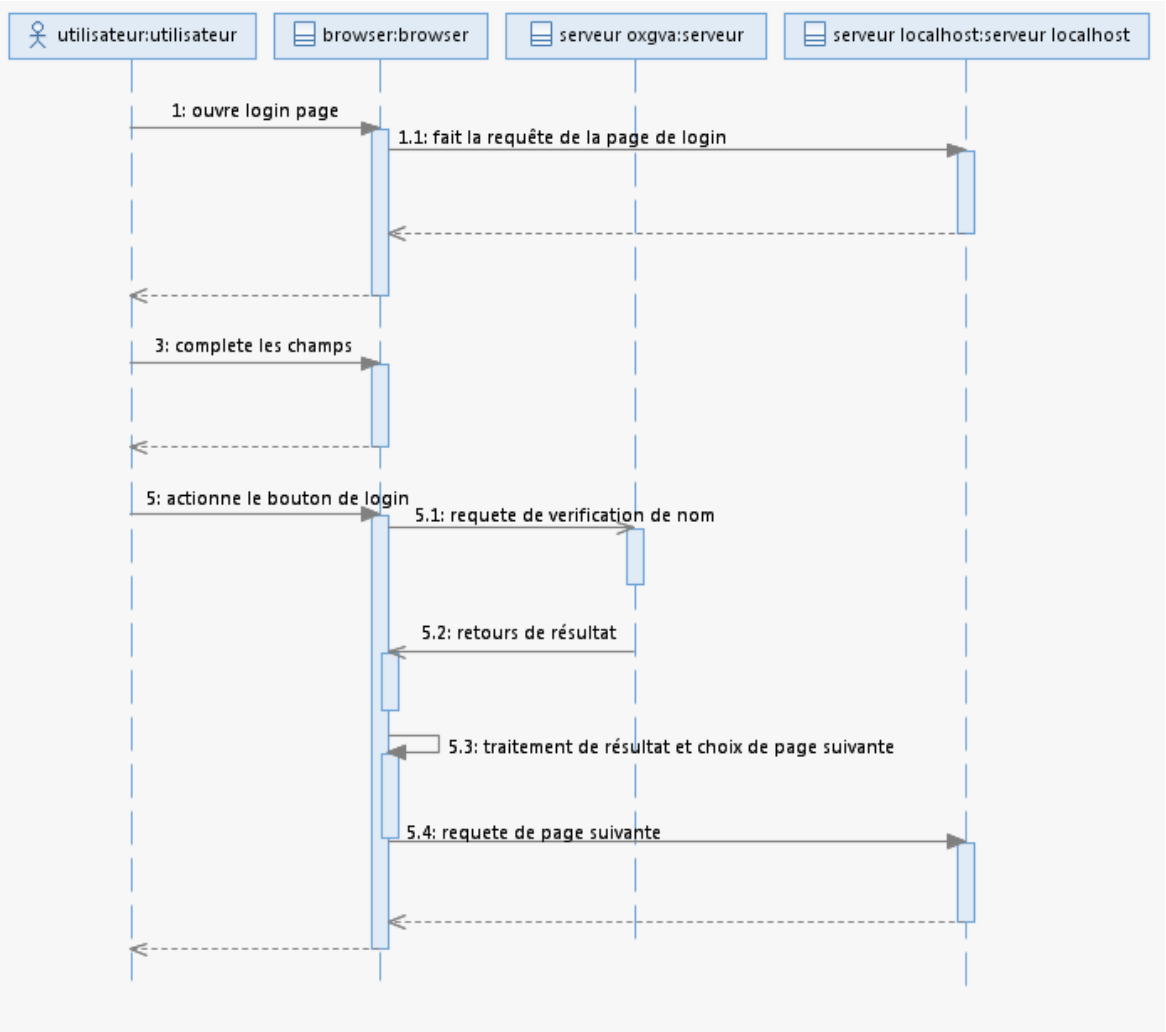
(Source : Marc Falcy)

Figure [25] : Architecture des serveurs concernés. Structure insuffisante



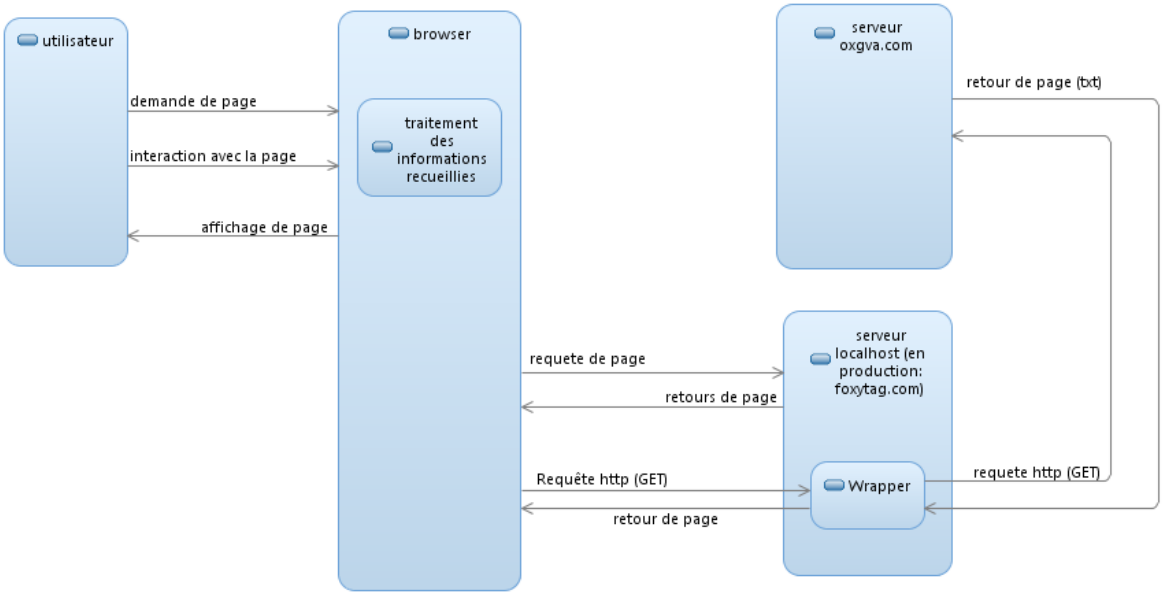
(Source : Marc Falcy)

Figure [26] : Séquence diagramme des requêtes. Structure insuffisante



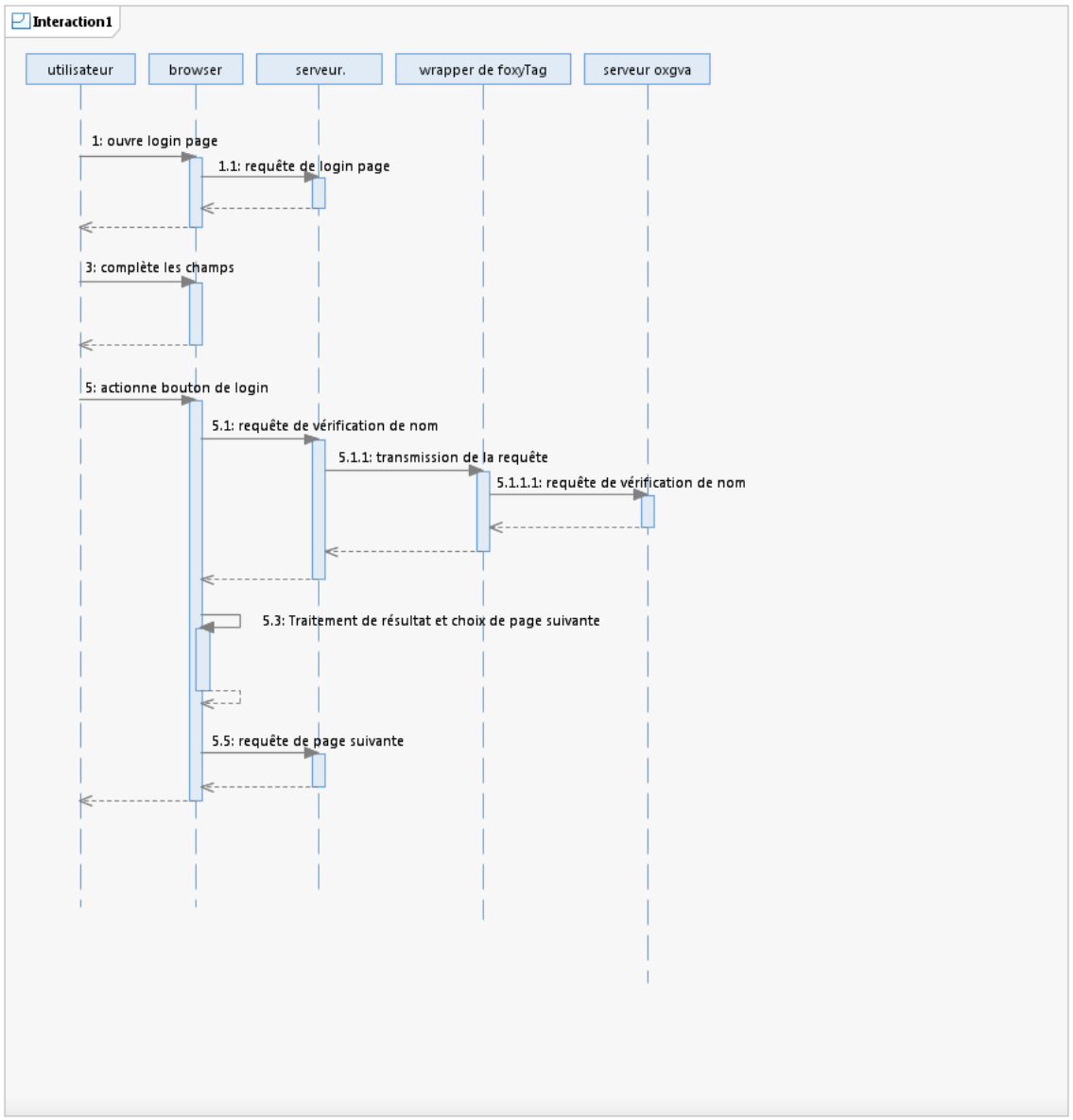
(Source : Marc Falcy)

Figure [27] : Architecture des serveurs concernés. Structure nécessaire



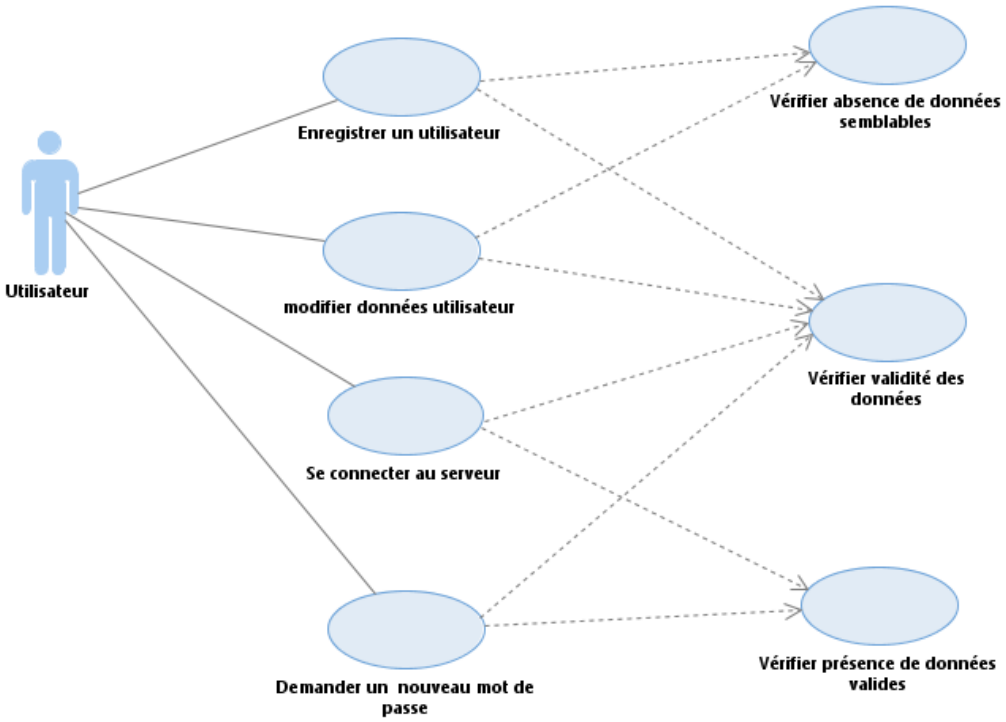
(Source : Marc Falcy)

Figure [28] : Séquence diagramme des requêtes. Structure nécessaire



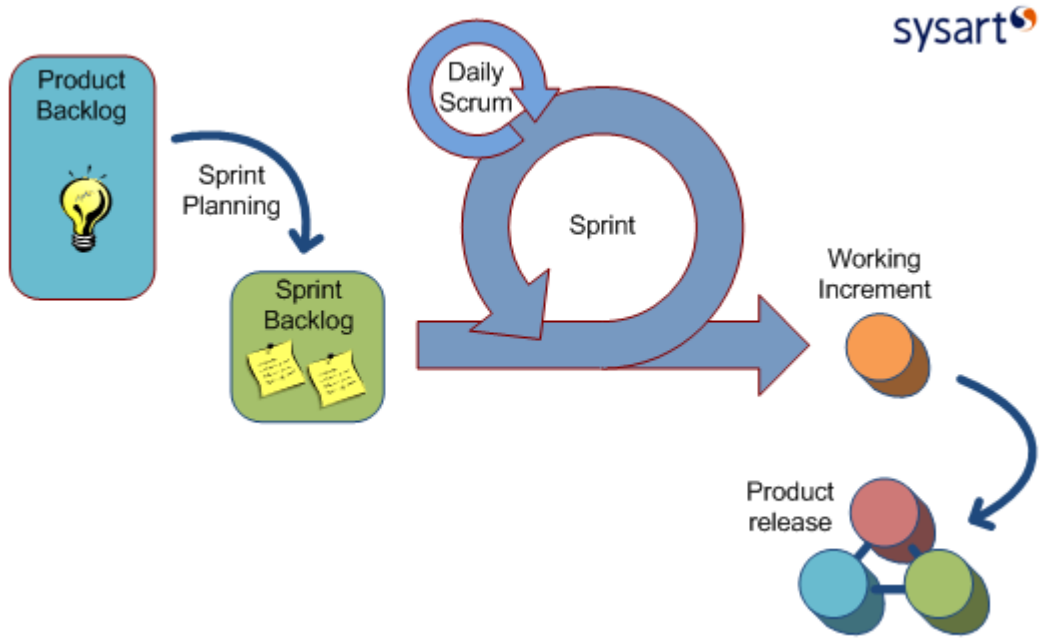
(Source : Marc Falcy)

Figure [29] : use-cases



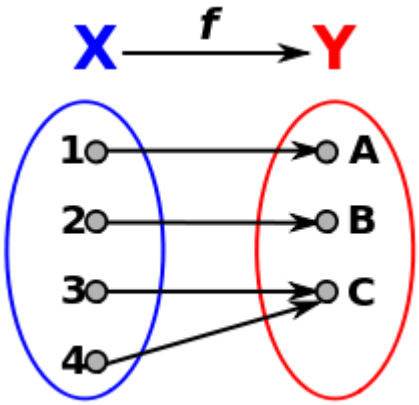
(Source : Marc Falcy)

Figure [30] : La méthode *SCRUM*



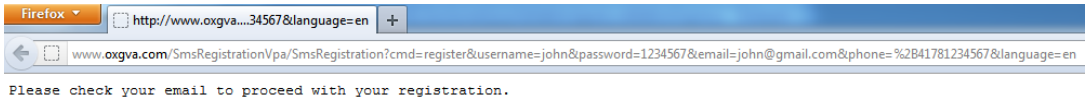
(Source: http://www.sysart.fi/news/31/37/Scrum-Resources/d,artikkeli)

Figure [31] : Non injectivité



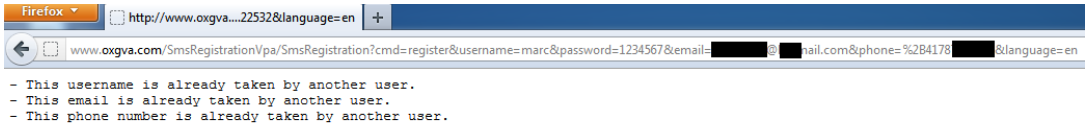
(Source: http://fr.wikipedia.org/wiki/Surjection)

Figure [32] : outil SmsRegistration cmd=register ok



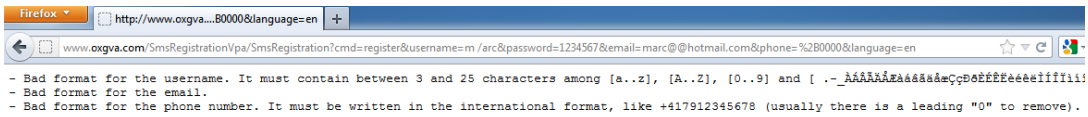
(Source : Marc Falcy)

Figure [33] : outil SmsRegistration cmd=register données déjà utilisées



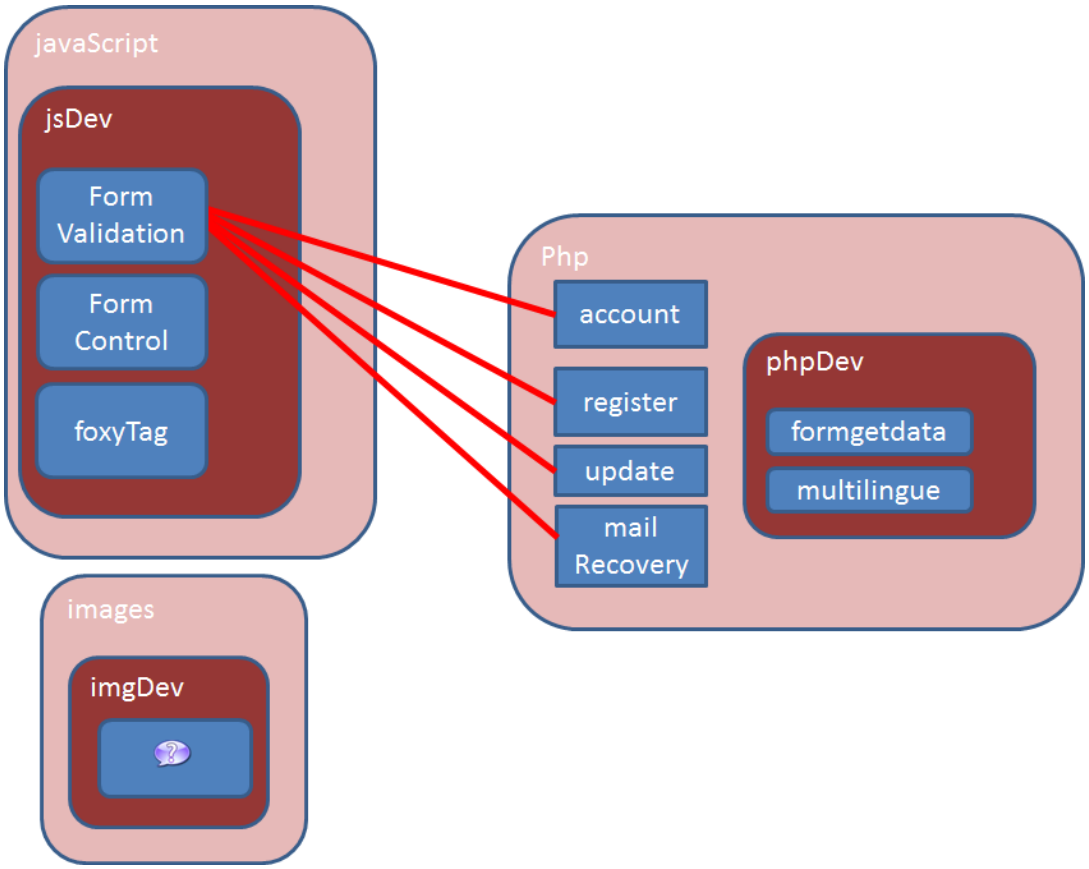
(Source : Marc Falcy)

Figure [34] : outil SmsRegistration cmd=register données invalides



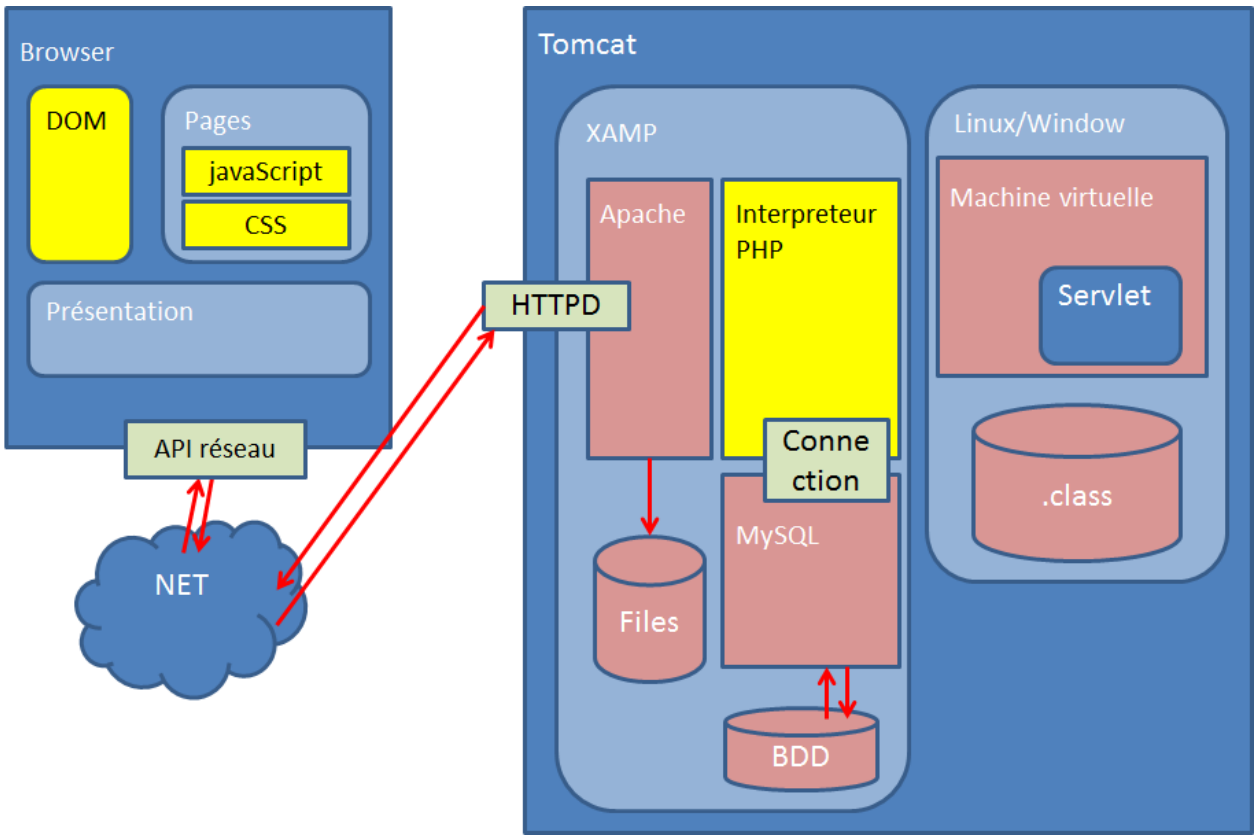
(Source : Marc Falcy)

Figure [35] : Structure centralisée



(Source : nouvelle architecture des pages foxytag. Version allégée par Marc Falcy)

Figure [36] : Architecture Serveur – Browser

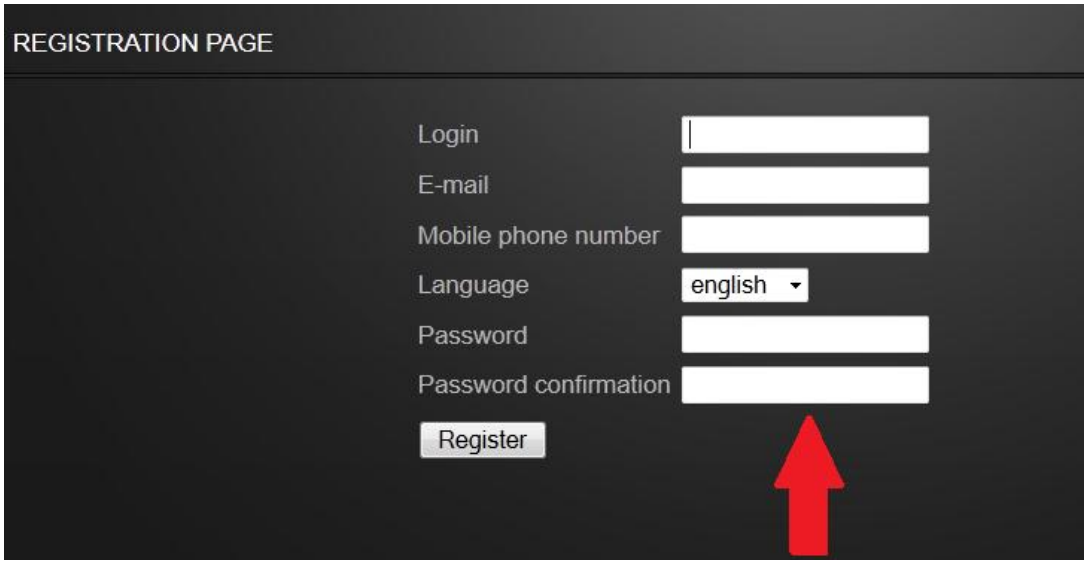


(Source : Johann Sievering)

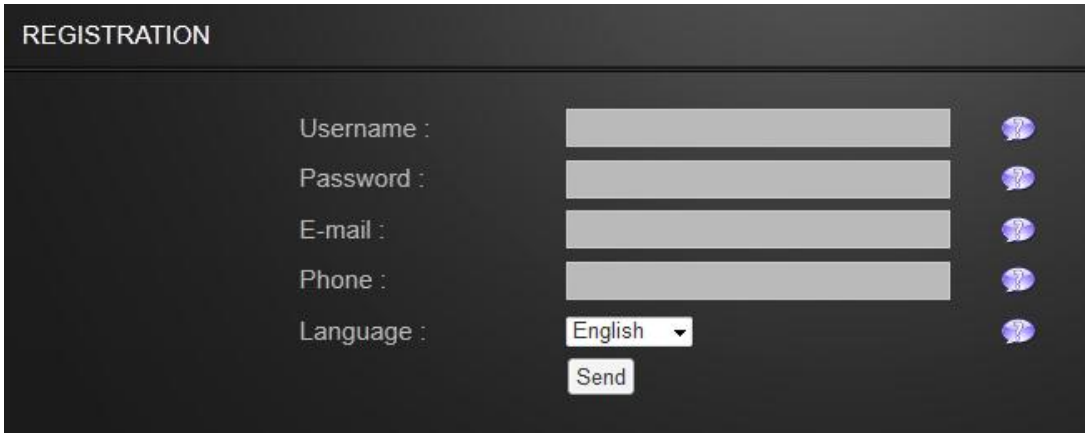
Figure [37] : Page de login

(Source : nouvel interface Web de foxytag par Marc Falcy)

Figure [38] : Page d'enregistrement

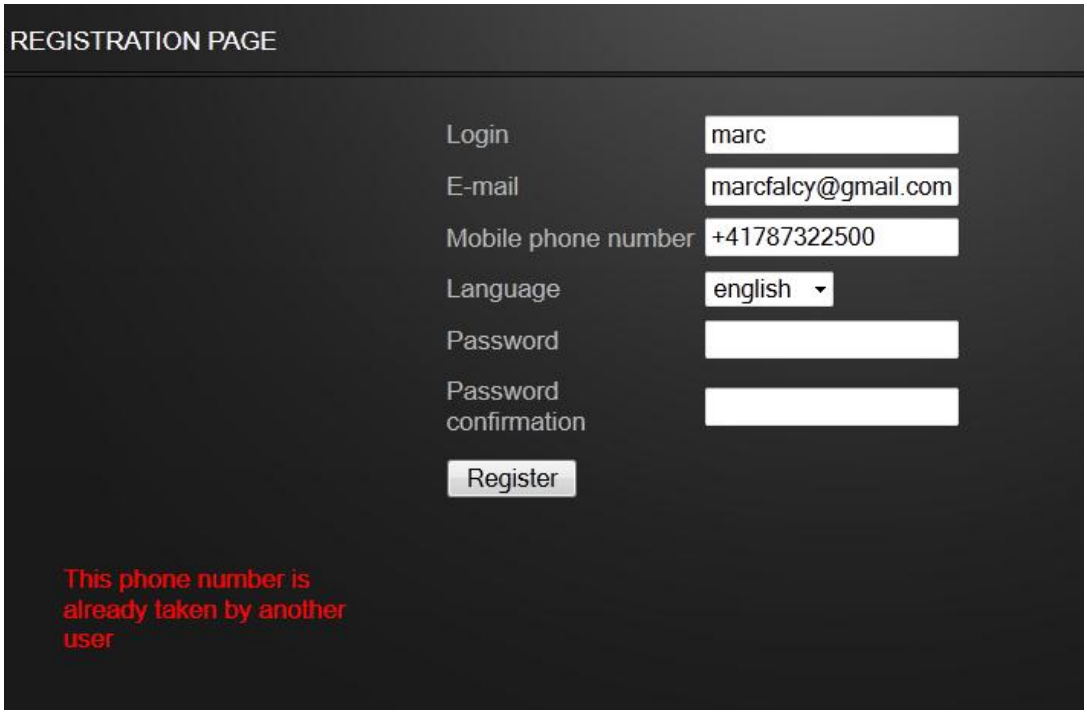


(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)



(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

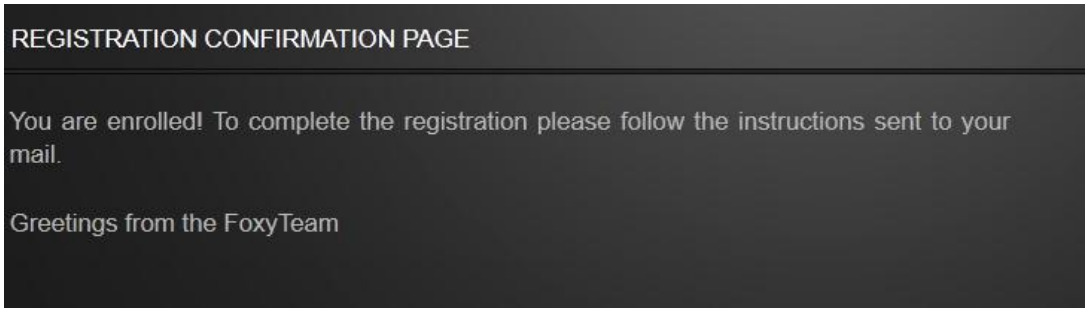
Figure [39] : Alerte de données déjà utilisées



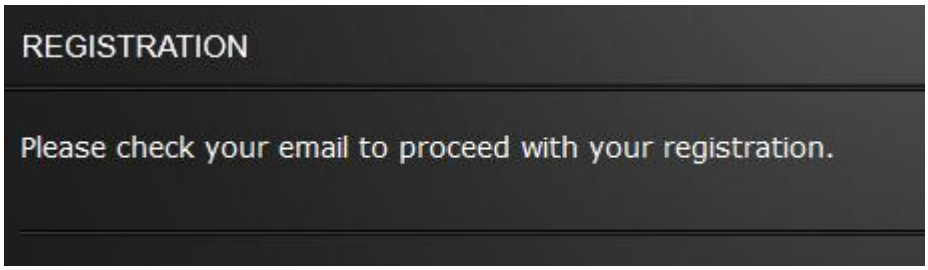
(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)

(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

Figure [40] : Confirmation d'enregistrement

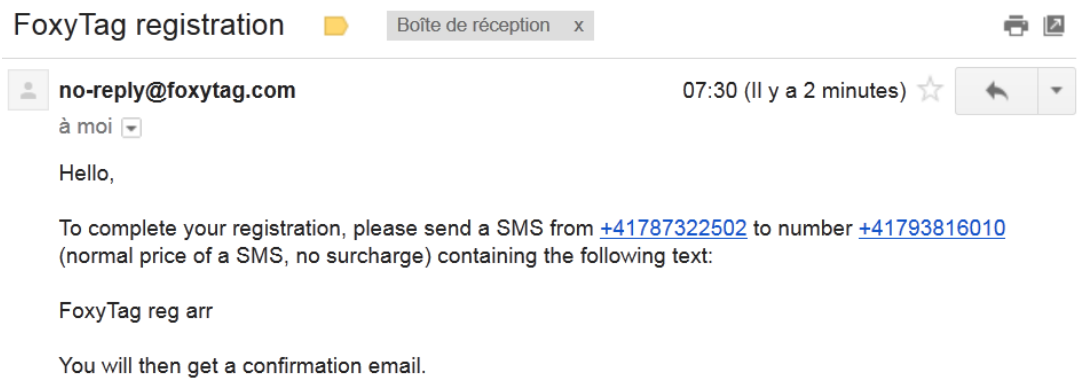


(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)



(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

Figure [41] : Demande de confirmation par mail


FoxyTag registration

Boîte de réception x

no-reply@foxytag.com 07:30 (Il y a 2 minutes)

à moi

Hello,

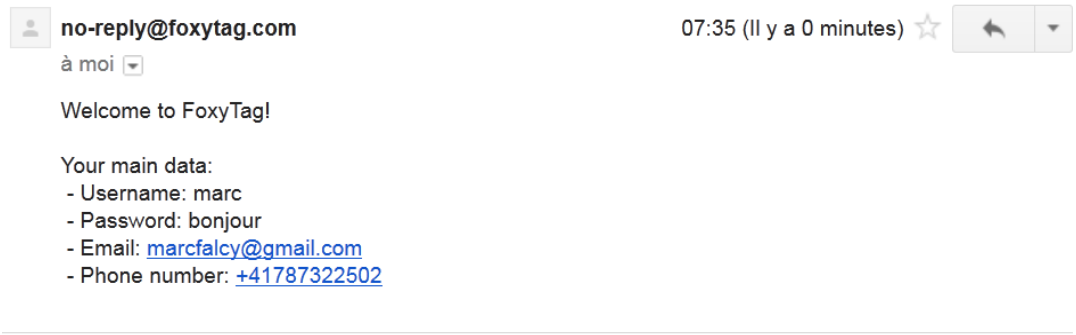
To complete your registration, please send a SMS from +41787322502 to number +41793816010 (normal price of a SMS, no surcharge) containing the following text:

FoxyTag reg arr

You will then get a confirmation email.

(Source : mail reçu du service de mailing de foxytag)

Figure [42] : Confirmation par mail


no-reply@foxytag.com 07:35 (Il y a 0 minutes)

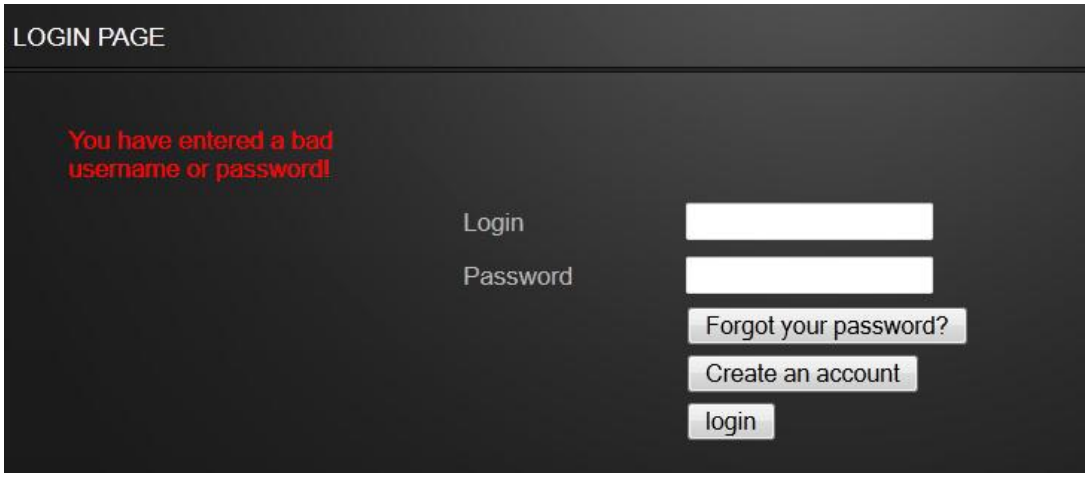
à moi

Welcome to FoxyTag!

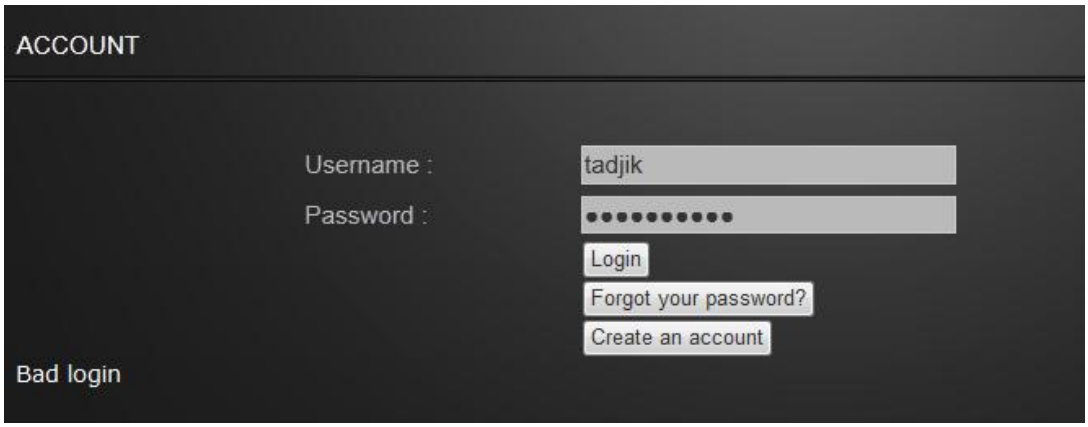
Your main data:
- Username: marc
- Password: bonjour
- Email: marcfalcy@gmail.com
- Phone number: +41787322502

(Source : mail reçu du service de mailing de foxytag)

Figure [43] : Annonce de mauvais login

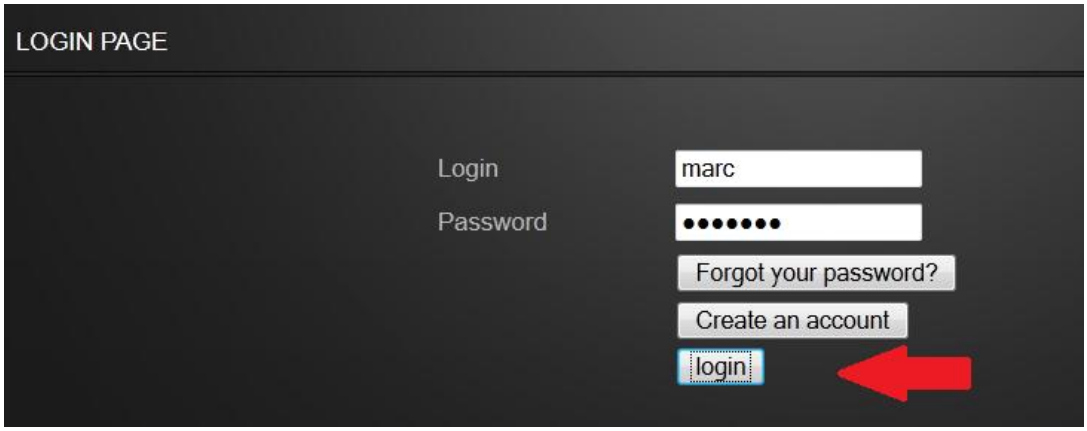


(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)



(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

Figure [44] : Clic pour se logger

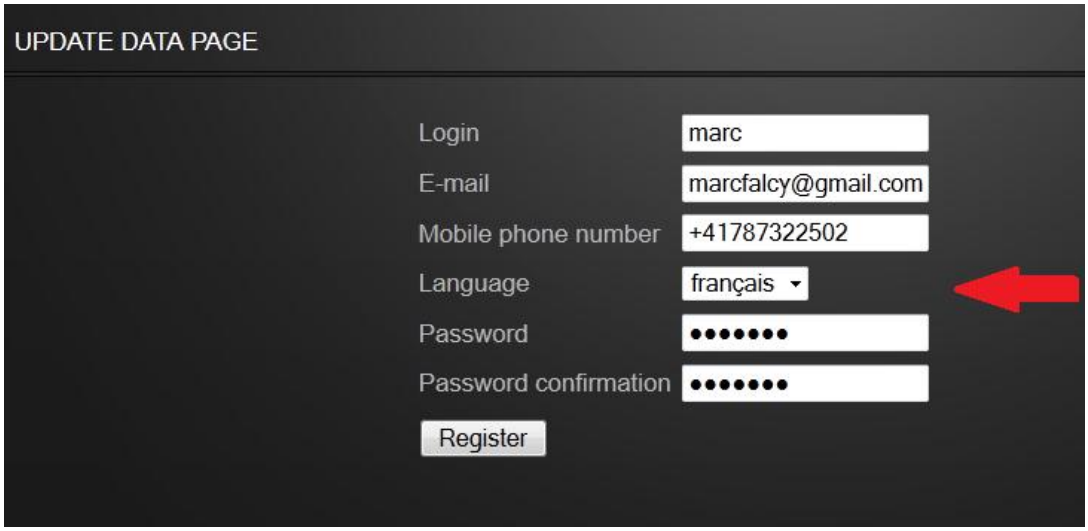


(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)

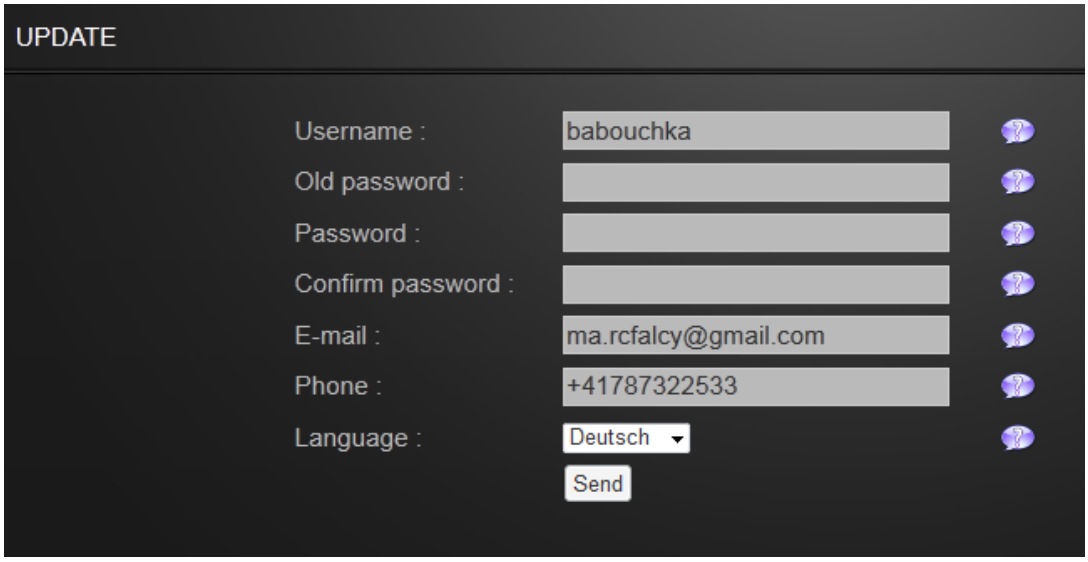
Figure [45] : Page d'un utilisateur

(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)

Figure [46] : Modification des données

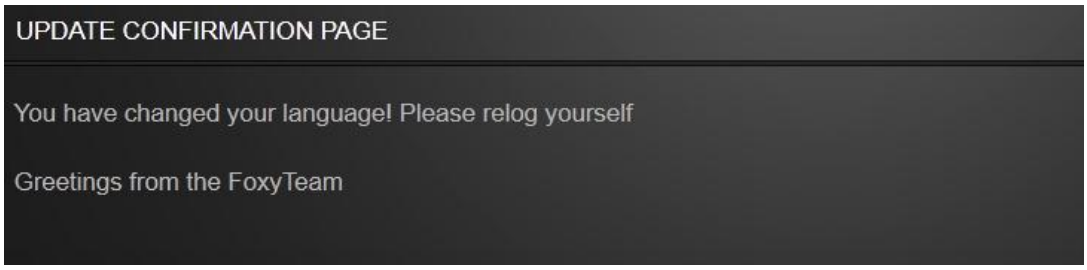


(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)



(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

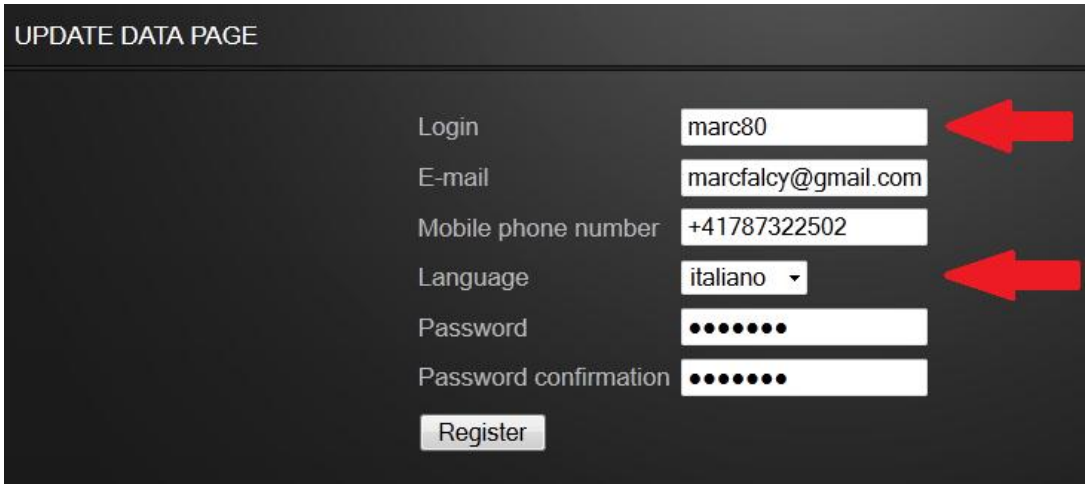
Figure [47] : Confirmation de modification de la langue



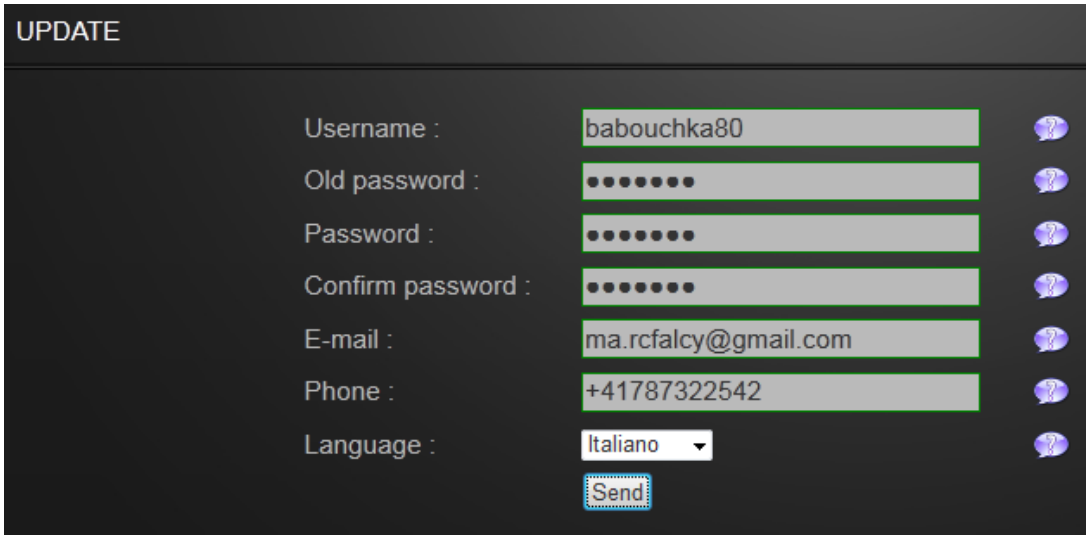
(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)

(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

Figure [48] : Page de modification des données

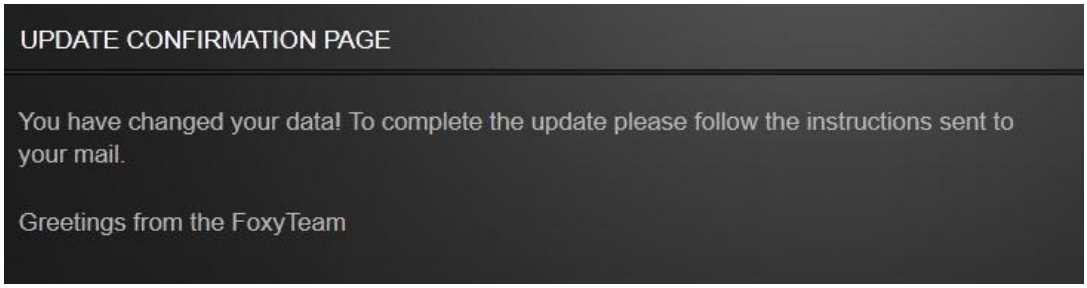


(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)

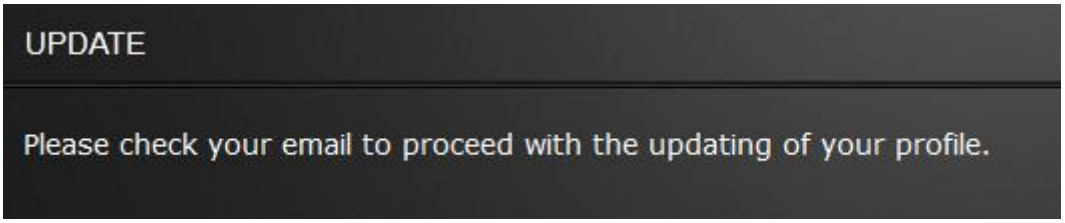


(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

Figure [49] : Confirmation de modification des données

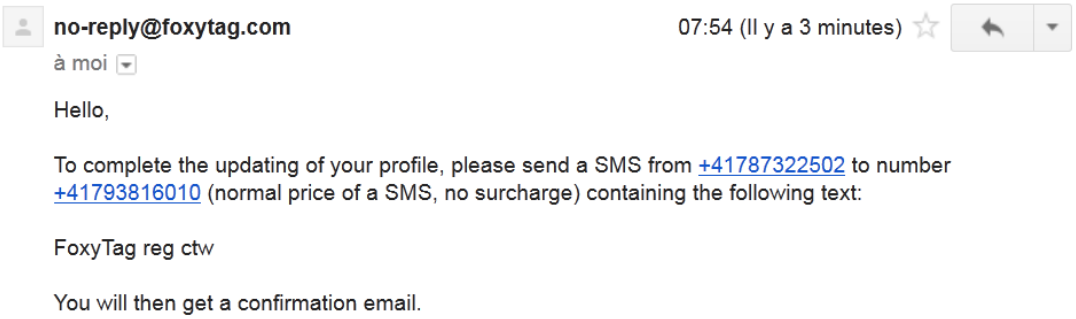


(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)



(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

Figure [50] : Demande de confirmation par mail



no-reply@foxytag.com 07:54 (Il y a 3 minutes)

à moi

Hello,

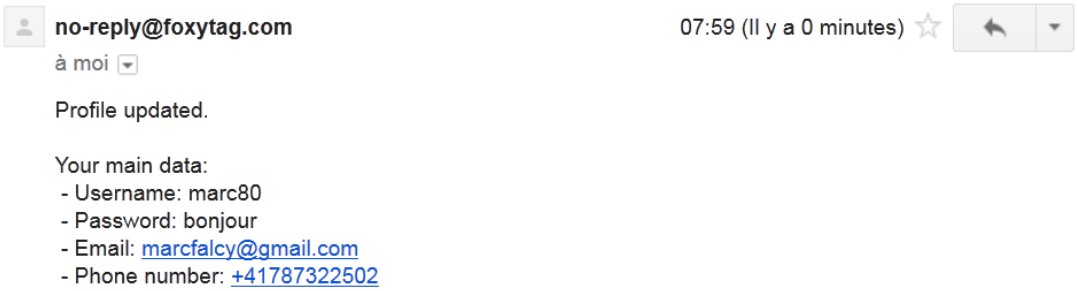
To complete the updating of your profile, please send a SMS from +41787322502 to number +41793816010 (normal price of a SMS, no surcharge) containing the following text:

FoxyTag reg ctw

You will then get a confirmation email.

(Source : mail reçu du service de mailing de foxytag)

Figure [51] : Confirmation par mail



no-reply@foxytag.com 07:59 (Il y a 0 minutes)

à moi

Profile updated.

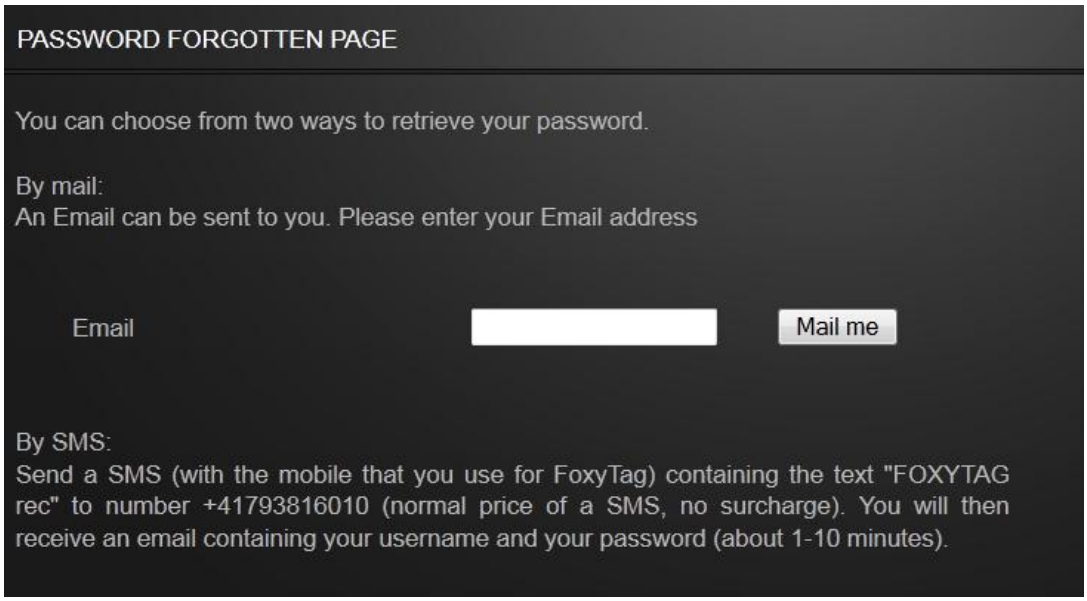
Your main data:
- Username: marc80
- Password: bonjour
- Email: marcfalcy@gmail.com
- Phone number: +41787322502

(Source : mail reçu du service de mailing de foxytag)

Figure [52] : Page de login

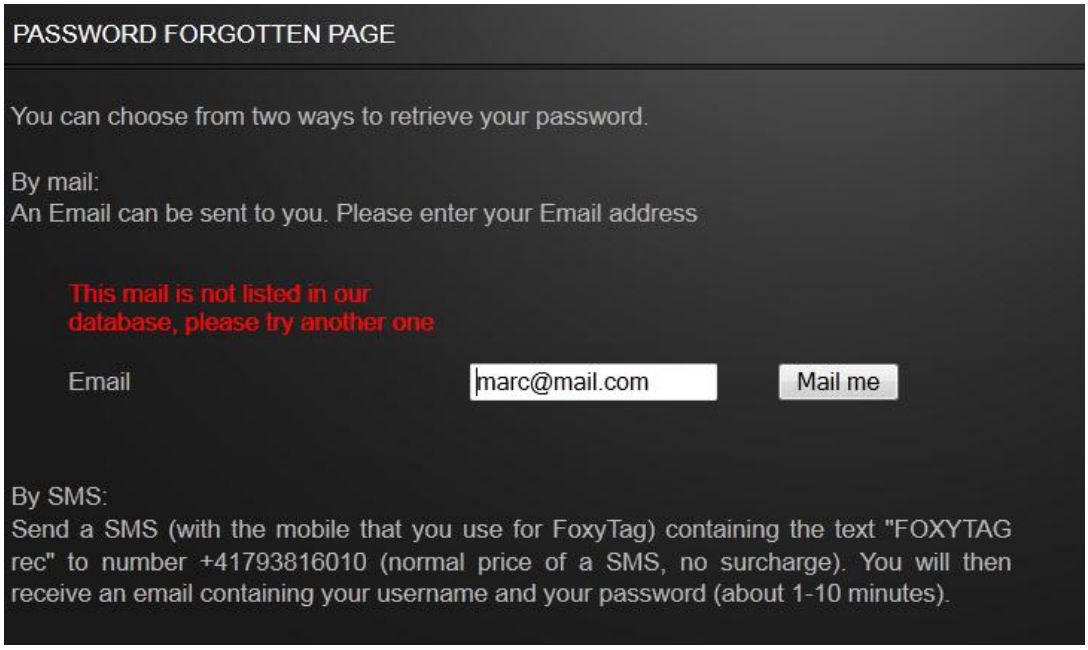
(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)

Figure [53] : Page de récupération de mot de passe

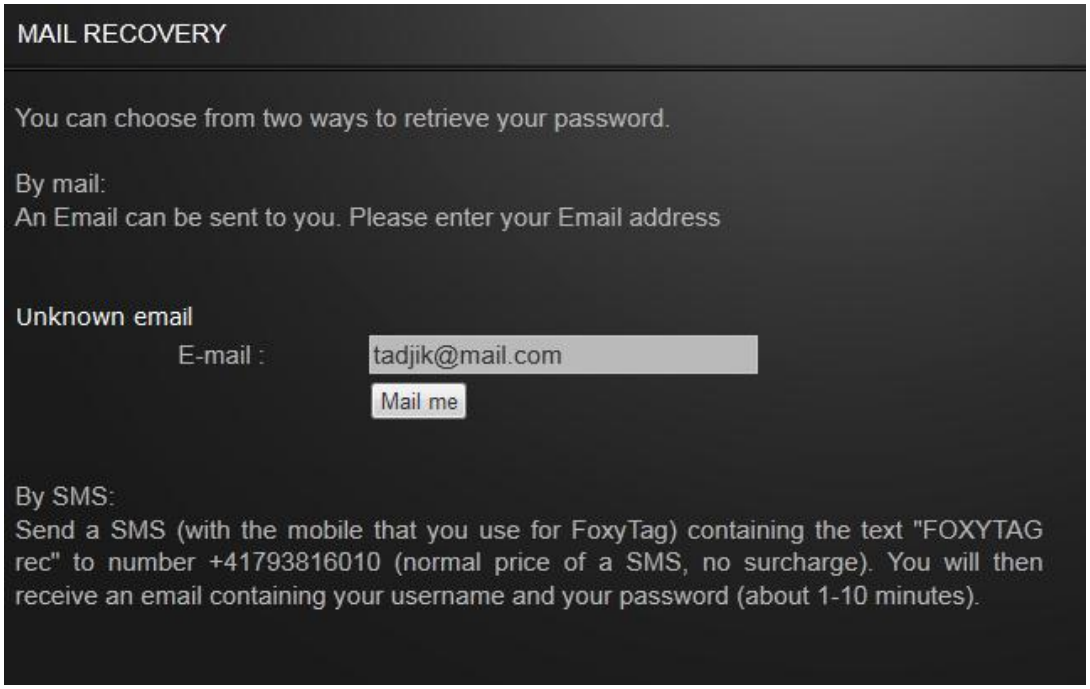


(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)

Figure [54] : Alerte d'adresse mail inexistante

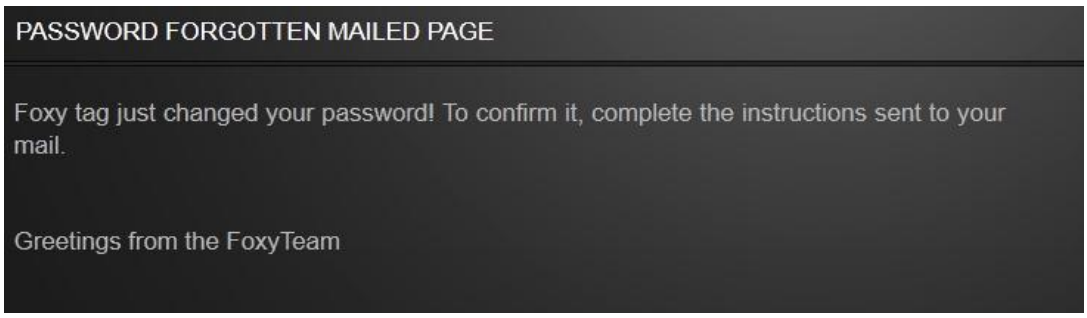


(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)

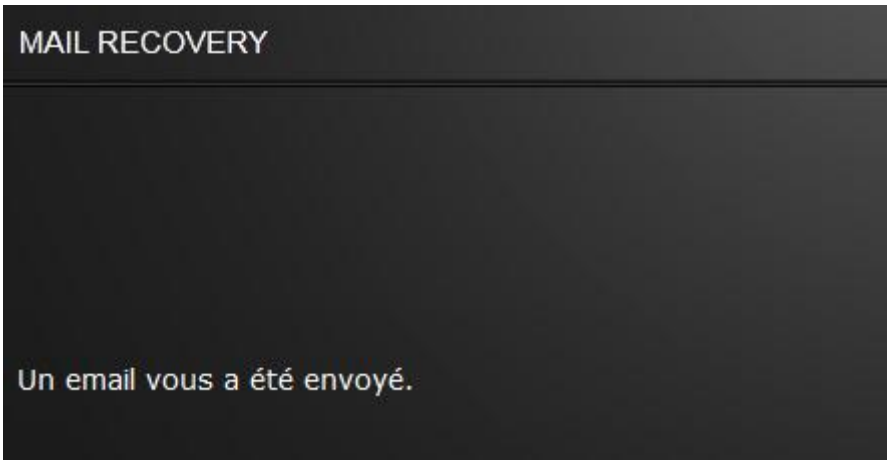


(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

Figure [55] : Confirmation d'envoi de mail



(Source : nouvel interface Web de foxytag. Version Ajax par Marc Falcy)



(Source : nouvel interface Web de foxytag. Version allégée, sans jQuery par Marc Falcy)

Figure [56] : Réception du nouveau mot de passe

(Source : mail reçu du service de mailing de foxytag)

Conclusion

Il ressort de ce travail de Bachelor que la méthode de gestion de projet est une méthode utile et très favorable pour des développeurs qualifiés travaillant dans un cadre de développement. Si ceux-ci ont une bonne connaissance et une bonne expérience dans leur domaine de compétences, ils pourront bien évaluer le temps de réalisation des logiciels ou des modules qu'ils développent. Par contre, s'il faut se former sur des outils pendant le développement du projet, il devient absolument impossible de prévoir correctement le temps de ce développement. Etant donné que le Scrum est basé sur une évaluation du temps des tâches à réaliser, il faut donc conclure que cette méthode particulière de gestion de projet n'est pas adéquate pour un étudiant en formation.

Le choix de ne pas utiliser le HTML5 ainsi que les capacités avancées de validation de formulaires a produit un rallongement du temps de développement. Ceci n'a pas permis d'effectuer de recherche sur les autres méthodes de connections d'utilisateurs, comme il était prévu dans le plan de réalisation. Cela dit, il faut prendre en compte la particulière valeur ajoutée aux connaissances des langages de développement web passant de faibles à intégrées. L'approfondissement détaillé et l'utilisation maîtrisée du HTML, du javaScript et de son extension jQuery, ainsi que d'autres outils comme le moteur de validation ou variables de session, autres tactiques de maîtrise du Web.

L'outil final pouvant être mis tel quel en production permettra au mandant de bénéficier d'une interface graphique rendant plus userfriendly l'inscription au compte. La valorisation de la non redondance du code PHP et javaScript en des fonctions spécifiques et établies pour elles-mêmes, ainsi que la limitation du nombre de pages à quatre, favorisera au maximum leur compréhension au moment de la maintenance et générera probablement un engouement pour la réutilisabilité.

Glossaire

Ajax :
Asynchronous Javascript And Xml. Permet d'actualiser le contenu d'une page, sans procéder au rechargement total de cette page. Un objet javaScript permet d'envoyé une requête http au serveur[39] .

Avertisseur de radar :
Système collaboratif permettant de fournir une indication sur la présence de radars routiers à des utilisateurs d'applications mobiles. Toléré dans certains pays, il est illégal dans d'autres.

Burndown chart :
Graphique indiquant la quantité de travail qu'il reste à effectuer. Moins il y a de tâches restant à réaliser, plus la courbe va descendre. Quand elle arrive à 0, le logiciel planifié est terminé.

Burnup chart :
Graphique indiquant la quantité de travail qui a été validé. Présente le désavantage de ne pas pouvoir considérer l'avancement, vu qu'il n'y a pas de zéro à atteindre. Normalement non utilisé pour la méthode de gestion *SCRUM.*

CSS :
Cascading Style Sheets. Langage de style qui définit la présentation de des documents HTML[40].

DFA :
Deterministic Finite Automation[41].

iFrame :
Une balise HTML insérée au code étant utilisée pour incorporer un autre document dans le document HTML courant[42].

Expressions régulières :
Abus d'utilisation de la langue anglaise, appelées « expressions rationnelles » en français.

---

[39] Source : http://www.futura-sciences.com/fr/definition/t/high-tech-1/d/ajax_3998/

[40] Source : http://fr.html.net/tutorials/css/lesson1.php

[41] Source : Regular Expression, Pocket Reference

[42] Source: http://www.w3schools.com/tags/tag_iframe.asp

Langage permettant de valider de manière absolue si une chaîne de caractères est contenue dans une autre.

HTML :

HyperText Mark-up Language. C'est un langage qui permet de présenter des informations sur Internet. Il utilise des balises pour définir les différent structures et composants du contenu d'une page[43].

Injectivité :

Une fonction est injective si et seulement si aucune image ne possède plusieurs préimages.

JavaScript :

Javascript est un langage de script orienté objet principalement utilisé dans les pages HTML. A l'opposé des langages serveur comme le PHP, le javaScript est exécuté sur le browser de l'utilisateur. Ainsi, ce langage permet une interaction avec l'utilisateur en fonction de ses actions et comportements[44].

jQuery :

jQuery est une bibliothèque javaScript rapide et concise, qui simplifie l'utilisation du code HTML. Elle peut gérer des événements, l'animation et les interactions Ajax pour le développement rapide et simplifié d'applications web[45].

Language:

Langue dans laquelle les notifications seront envoyées à l'utilisateur.

Little day :

Un jour de travail complet - temps quotidien d'organisation

Mail:

Adresse mail de l'utilisateur de foxyTag. Cette valeur ne doit être présente que pour une seule personne sur la base de données oxgva.

NFA :

Non deterministic Finite Automation[46].

---

[43] Source: http://fr.html.net/tutorials/html/lesson2.php

[44] Source: http://www.futura-sciences.com/fr/definition/t/internet-2/d/javascript_509/

[45] Source: http://jquery.com/

[46] Source : Regular Expression, Pocket Reference.

Password:
Chaîne de caractères associée au username permettant de valider l'identification d'un utilisateur. Elle est dans le cadre de foxyTag codée avec l'algorithme de hachage SHA-1. Pour assurer la sécurité des données de l'utilisateur, elle devrait rester connue de lui seul.

Phone:
Numéro de téléphone de l'utilisateur sur le site foxyTag. C'est un numéro unique et personnel.

PHP :
PHP est un acronyme récursif, signifiant *PHP Hypertext Preprocessor*. C'est un langage de script HTML, exécuté côté serveur[47].

Planning Poker:
Action où chacun des membres de l'équipe fourni une estimation du temps que va prendre une tâche. En plus d'être très utile pour comprendre chaque tenant et aboutissant de la tâche, puisque l'avis de chacun des membres de l'équipe est pris en compte, c'est un moment particulièrement ludique.

Product Backlog:
Ensemble des tâches définies pour la réalisation d'un projet. Leur définition se fait en tout début de projet[48].

Product Owner:
Personne qui à la connaissance de l'entreprise pour laquelle le produit est développé. Elle participe au projet[49].

***Requête GET* :**
Requête HTTP. Méthode courante pour demander une ressource à un serveur. Elle est sans effet sur la ressource. Il devrait être possible de la répéter sans effet. Bien que ce ne soit pas le cas dans ce travail[50].

SCRUM :
Méthode de gestion de projet spécifiquement utile au développement d'applications informatiques[51].

---

[47] Source : http://www.php.net/manual/fr/preface.php

[48] Cf. SCRUM

[49] Cf. SCRUM

[50] Source : http://www.siteduzero.com/tutoriel-3-35613-les-requetes-http.html

[51] Source : Cours du professeur Dugerdil. HEG, semestre 5

SHA-1 :
Algorithme de hachage.

Sprint:
Incrément itératif de la méthode.

Sprint Backlog:
Somme des fonctionnalités décidées en début de projet. La réalisation de tâches mettant en œuvre ces fonctionnalités permet de remplir le cahier des charges du développement de l'application[52].

TaM:
Transport and Mobility

Username:
chaîne de caractères identifiant l'utilisateur sur le site de foxytag.com. Elle est unique et personnelle.

Wrapper :
Code PHP permettant de retransmettre une requête Ajax reçue d'un browser pour la transmettre à un autre serveur. Le wrapper doit être intégré à un serveur. Etant donné que les standards de sécurité actuels permettent les appels Ajax « serveur demandeur » - « serveur interrogé », mais ne les autorisent pas pour les appels « browser » - « serveur interrogé », il faut engendrer un pont de communication « browser » - « serveur demandeur » (wrapper) - « serveur interrogé ».

[52] Source : Cours du professeur Dugerdil. HEG, semestre 5

Bibliographie

- Stubblebine, T., *Regular Expression, Pocket Reference,* Editions O'Reilly.
- Friedl, J, *Mastering Regular Expression,* Editions O'Reilly.

Webographie

- Gestion d'utilisateur: http://oracle.developpez.com/guide/administration/adminuser/#L1.1.1
- Méthodes d'authentification : http://matthieu.developpez.com/authentification/
- Authentification d'un utilisateur dans un Framework : http://framework.zend.com/manual/fr/learning.multiuser.authentication.html
- Inscription de membres en utilisant du PHP : http://forum.phpfrance.com/faq-tutoriels/inscription-connexion-dans-espace-membres-t242539.html
- Statistique de browsers : http://www.w3schools.com/browsers/browsers_stats.asp
- Exemple de service de validation : http://validator.w3.org
- Moteur de validation utilisé pour le prototype : http://www.position-absolute.com/articles/jquery-form-validator-because-form-validation-is-a-mess/
- Vérificateur d'expressions rationnelles : http://gskinner.com/RegExr/
- Précision des spécifications du Scrum : http://en.wikipedia.org/wiki/Scrum_%28development%29
- Support au javaScript : http://books.lifeleaks.com/eloquentjavascript/contents.html
- Support à la validation avec jQuery : http://www.chezneg.fr/leblog/chezneg-leblog.php?id_art=126
- Support à la validation et aux fonctions Ajax : http://www.chezneg.fr/leblog/chezneg-leblog.php?id_art=220
- Support aux fonctions Ajax avec jQuery : http://www.snoupix.com/initiation-a-ajax-avec-jquery-partie-1_tutorial_20.html
- Support aux fonctions jQuery : http://stackoverflow.com/questions/203844/jquery-validation-plugin-disable-validation-for-specified-submit-buttons
- Support aux fonctions jQuery : http://docs.jquery.com/Ajax_Events

- Support aux fonctions Ajax :
  http://blog.pascal-martin.fr/post/prototype-gestionnaires-evenements-pour-Ajax-Request
- Rôle de Scrum :
  http://freecode.com/articles/an-introduction-to-scrum
- Compréhension des problèmes liés au Cross-Domain :
  http://www.siteduzero.com/tutoriel-3-56320-l-xmlhttprequest-cross-domain.html
- Module Cross-Domain jQuery :
  https://github.com/padolsey/jQuery-Plugins/tree/master/cross-domain-ajax/