

L'ergonomie dans la conception d'un formulaire

(date de rédaction : 24.07.2004)

Sommaire

Introduction

1. Le choix et la spécification des éléments de formulaire
2. Organisation visuelle du formulaire
3. Rapidité d'exécution
4. Sûreté d'exécution : la gestion des erreurs

Conclusion

Pour en savoir plus...

Référence

<http://www.ergolab.net/articles/ergonomie-conception-formulaire.html>

Introduction

Le formulaire est l'un des moyens pour l'utilisateur d'envoyer des données, alors que son comportement classique sur le web consiste plutôt à recevoir des informations. Cette notion entraîne des problématiques d'ergonomie autres que celles liées à la simple consultation de contenus.

Cet article présente les standards d'ergonomie web pour la conception de formulaires et quelques recommandations à adapter à la spécificité de chaque situation. Les thématiques concernant les formulaires couvrent un large champ d'application, dans les domaines du logiciel et du web, pour le grand public ou les applications métiers, pour les novices ou les utilisateurs expérimentés.

Qu'ils servent à envoyer un message, acheter un produit, remplir un questionnaire ou accomplir une tâche professionnelle plus précise, les formulaires doivent respecter certaines règles. On peut distinguer des recommandations génériques et des recommandations spécifiques, qui seront déduites du contexte d'utilisation du formulaire et de ses caractéristiques.

Il existe une grande différence entre un formulaire destiné à un site grand public ou à une application professionnelle. Cette différence est d'ailleurs à étudier quelque soit la thématique (mode d'interaction avec le formulaire, organisation des champs, gestion des erreurs, etc.).

Les formulaires sont quelque chose d'assez bien défini au niveau des règles à appliquer pour les formulaires web classiques (formulaire de contact, de commande d'un document, de gestion d'un processus d'e-commerce...).

C'est beaucoup plus complexe pour les formulaires spécifiques, destinés à être utilisés dans un cadre professionnel, et pour lesquels on ne dispose pas de référent existant. Pour ce type de formulaires, l'analyse de l'activité doit prendre une place beaucoup plus importante, c'est elle qui doit guider les choix d'interface plutôt que l'application de règles standards.

1. Le choix et la spécification des éléments de formulaire

La première étape dans la conception d'un formulaire consiste à déterminer son utilité, ce à quoi il doit servir, et les informations qu'il doit permettre de recueillir. Les méthodes d'analyse de l'activité et des supports existants ainsi que les méthodes d'interview permettent de compléter cette première étape.

Après avoir décidé des informations à recueillir via le formulaire, il faut trouver la manière optimale de les demander à l'utilisateur. Un formulaire se décompose en différents éléments de formulaire. Les caractéristiques de chaque élément permettent de récupérer des types d'informations particuliers.

■ La ligne

L'élément "Ligne" permet d'obtenir un champ de formulaire d'une seule ligne. Les paramètres qui peuvent être spécifiés sont la longueur de la ligne et les données qui peuvent y être entrées. Les valeurs de ces paramètres doivent être adaptées au type de données afin de faciliter le renseignement du formulaire par les utilisateurs :

E-mail

ISBN 2910565092

Exemple : les deux champs lignes "E-mail" et "ISBN" n'ont pas la même taille du fait de la différence des informations attendues.

■ Le champ texte

L'élément "Champ texte" permet d'obtenir un champ de formulaire d'une ou plusieurs lignes. Les paramètres qui peuvent être spécifiés sont le nombre de lignes, la longueur de la ligne et les données qui peuvent y être entrées. Les valeurs de ces paramètres doivent être adaptées au type de données.

■ Le bouton radio

L'élément "Bouton radio" correspond à un élément de formulaire permettant une sélection unique. Les boutons radio sont donc une série de boutons exclusifs, fonctionnant sur le mode "On - Off". Ils permettent de présenter des réponses alternatives à une question:



Exemple d'utilisation de boutons radio : choix entre l'option "Oui" et l'option "Non".

L'utilisation des libellés "Oui" et "Non" permet d'insister sur le caractère binaire de cet élément de formulaire. Il est plus indiqué pour une utilisation de formulaire par les novices, mais nécessite plus d'espace à l'écran qu'une simple case à cocher.

Les avantages des boutons radio (ainsi que des cases à cocher) tiennent essentiellement au fait que toutes les options sont visibles sans nécessiter d'action de la part de l'utilisateur (contrairement à une liste déroulante par exemple). Le pendant de cet avantage est que ce type d'élément de formulaire prend beaucoup de place à l'écran et peut compliquer la lecture du formulaire.

Les boutons radio sont des éléments qui ne peuvent fonctionner qu'en groupe (c'est à dire au moins deux boutons radio). Il ne peut donc pas exister d'interface avec un seul bouton radio.

Le bouton radio doit être placé par défaut sur l'option la plus fréquemment choisie, ou que l'on suppose la plus appropriée en fonction d'une analyse de la tâche. S'il n'existe aucune option préférentielle, les boutons radio peuvent être laissés décochés. Cependant, cela oblige l'utilisateur à accomplir une action de sélection quelque soit la situation.

La présélection par défaut a un inconvénient lorsque la sélection d'un bouton radio est optionnelle. En effet, une fois que l'utilisateur a sélectionné une option, il ne peut pas revenir en arrière sauf si le concepteur a prévu un bouton "pas de sélection".

■ La case à cocher

L'élément "Case à cocher" permet d'obtenir une liste d'options au sein de laquelle la multisélection est possible. La case à cocher est aussi employée pour des choix binaires (le cochage / décochage de la case active ou désactive une fonction).

Les cases à cocher peuvent être présentées seules ou en groupe. Lorsqu'elles fonctionnent en groupe, les choix qu'elles proposent sont non-exclusifs. Les cases à cocher comme les boutons radio fonctionnent sur le mode "On - Off" (on coche ou on décoche la case). Les avantages de la case à cocher sont identiques à ceux des boutons radio.

- Vendredi
- Samedi
- Dimanche

Exemple d'utilisation de cases à cocher. La présence des 3 cases à cocher (Vendredi, Samedi et Dimanche) permet à l'utilisateur d'effectuer :

- Une sélection unique (une option parmi les 3 proposées)
- Une multisélection (2 ou 3 options parmi les 3 proposées)

■ Le menu déroulant

L'élément "Menu déroulant" permet d'obtenir une liste d'options avec un libellé visible (soit l'intitulé du champ, soit la première option de la liste, soit une option sélectionnée par défaut). Ce type d'élément d'interface permet de proposer un grand nombre d'options sans prendre beaucoup de place à l'écran. L'utilisateur ne peut faire que des choix uniques.

Rubrique

Exemple d'utilisation d'un menu déroulant : le menu déroulant intitulé "Rubrique" permet de choisir à quelle rubrique appartient un article d'Ergolab. Ce type d'interface permet de proposer 4 options sur une seule ligne à l'écran (à l'inverse de cases à cocher où toutes les options sont visibles à l'écran).

■ La liste

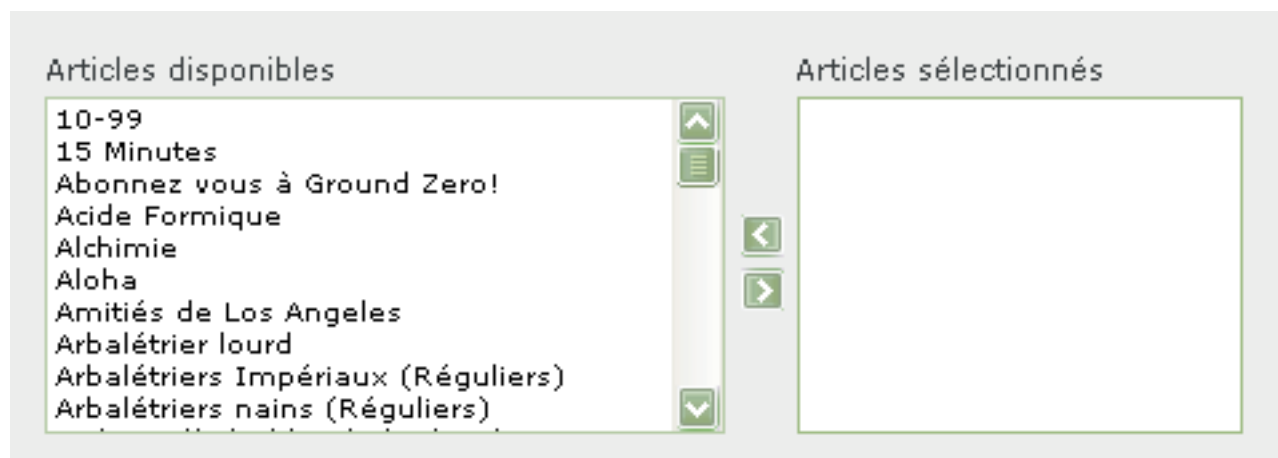
L'élément "Liste" permet d'obtenir une liste d'options avec la liste des premières options visible à l'écran (en fonction de la longueur visible de la liste à l'écran). L'utilisateur peut en général effectuer des choix multiples, sauf si le concepteur a spécifié l'inverse.

Articles disponibles

10-99
15 Minutes
Abonnez vous à Ground Zero!
Acide Formique
Alchimie
Aloha
Amitiés de Los Angeles
Arbalétrier lourd
Arbalétriers Impériaux (Réguliers)
Arbalétriers nains (Réguliers)

■ La double liste

L'élément "Double liste" permet d'obtenir deux listes d'options avec possibilité de passage d'éléments d'une liste à une autre.



2. Organisation visuelle du formulaire

2.1. Présentation générale du formulaire

La conception d'un formulaire doit avoir pour objectif de permettre aux utilisateurs de le renseigner rapidement et efficacement (avec le moins d'erreurs possibles).

Le type de présentation que l'on choisit doit être adapté aux caractéristiques de l'utilisateur et de la tâche : est-ce que les utilisateurs connaissent bien l'outil informatique, est-ce qu'ils utilisent fréquemment le formulaire (et à quelle fréquence), est-ce que la rapidité d'exécution est capitale, etc.

Un formulaire peut être présenté sur une seule page, ou découpé en plusieurs pages. Cette dernière configuration est appelée "wizard" en raison du caractère "assisté" du remplissage du formulaire : on guide l'utilisateur à travers plusieurs étapes successives, jusqu'à ce qu'il ait renseigné toutes les informations dont on a besoin.



WIZARD



Plusieurs pages successives permettent de renseigner toutes les informations

- **AVANTAGES**
 - Simplicité de chacune des pages
 - Gestion des erreurs simplifiée
 - Gestion des dépendances entre champs
- **DEFAUTS**
 - Lenteur d'exécution de la tâche
 - Difficulté à juger la durée de renseignement

PAGE UNIQUE



Une seule page permet de renseigner toutes les informations

- **AVANTAGES**
 - Rapidité d'exécution de la tâche
- **DEFAUTS**
 - Complexité de la page
 - Gestion des dépendances entre champs

Chacune des configurations a des avantages et des défauts, qui sont plus ou moins importants en fonction du type d'application, du niveau d'expertise de la cible utilisateur et de sa fréquence d'utilisation du formulaire. Certains inconvénients peuvent être contournés en adoptant des solutions simples.

■ Wizard

» Avantages :

- Simplicité de chacune des pages : puisque le formulaire est découpé en plusieurs parties, on ne présente à l'utilisateur qu'une partie des informations sur chaque page. Ce procédé permet qu'il se concentre sur les renseignements qu'on lui demande de fournir, sans être distrait par d'autres groupes de champs du formulaire. Le renseignement d'un formulaire par étapes successives permet de réduire la complexité perçue de la procédure.
- Gestion des erreurs simplifiée : lorsque l'utilisateur renseigne mal une des pages (oubli d'informations obligatoires ou mauvais format de réponse), on lui affiche un message et on l'aide à corriger ses erreurs. Ce type d'informations est beaucoup plus "digestible" lorsque la page est de taille réduite que lorsque l'on a affaire à un long formulaire. En effet, la détection du ou des champs mal renseigné(s) est beaucoup plus aisée. Découper le formulaire en plusieurs pages permet donc d'optimiser le rapport signal / bruit en faveur d'une détection rapide : on repère plus facilement un message d'erreur parmi 5 champs que parmi 35.

- Gestion des dépendances entre champs : si la présence de certains champs ou leurs valeurs par défaut dépendent de champs précédents, il est plus facile de le gérer lorsque le formulaire est divisé en plusieurs étapes. Cela permet à l'utilisateur de ne pas se poser de questions sur ce qu'il a déjà renseigné pour pouvoir renseigner les champs suivants.

» **Défauts :**

- Lenteur d'exécution de la tâche : chaque étape demande à être validée avant de pouvoir passer à la suivante. De plus, la validation des champs doit se faire à chaque étape, alors que dans un formulaire unique l'utilisateur peut corriger toutes ses erreurs dans la même "session" de travail, sans recharger la page.

- Difficulté à juger de la durée de renseignement : l'utilisateur qui remplit la première page d'un wizard n'a pas a priori d'informations sur la longueur totale du wizard. Il peut donc difficilement opérer des choix stratégiques (du style "Je sais que ce wizard est composé de 10 pages, j'ai un rendez-vous dans 5 minutes, je choisis de remplir ce wizard plus tard"). La parade consiste à fournir des numéros de page accompagnés du nombre total de pages composant le formulaire.

■ Page unique

» **Avantages :**

- Rapidité d'exécution de la tâche. Le passage d'un champ au suivant peut toujours se faire au clavier et très rapidement, sans rechargement de la page (alors que dans un wizard, chaque passage à la page suivante demande du temps).

» **Défauts :**

- Complexité de la page : puisqu'on doit demander toutes les informations sur la même page, on a nécessairement une interface qui visuellement est plus chargée que dans un wizard. Cette charge informationnelle peut perturber l'utilisateur dans sa lecture du formulaire.

- Gestion des dépendances entre champs : lorsque la présence de certains champs ou leurs valeurs par défaut dépendent de champs précédents, un formulaire unique n'est pas la solution la plus adaptée. Elle nécessite des actions du système ou de l'utilisateur:

- Actions du système : le formulaire se recharge ou s'actualise en fonction des données entrées

- Actions de l'utilisateur : l'utilisateur doit faire une sélection mentale des champs qui s'appliquent à sa situation

2.2. Groupement et ordonnancement des champs

Un travail très important lors de la conception d'un formulaire consiste à rendre l'interface claire et lisible. Cette tâche comprend notamment des actions de groupement et d'ordonnancement des champs.

Il s'agit de choisir quels champs "vont ensemble", et dans quel ordre on affiche ces champs et les groupes de champs. On devra donner à chacun des groupes de champs un titre.

On recommande en général dans la conception des formulaires de se baser sur les habitudes des utilisateurs dans le "monde réel" (par exemple pour des opérateurs qui saisissent des commandes de vente par correspondance, le formulaire informatique doit ressembler au formulaire papier envoyé par les clients). Cependant, il faut éviter de reproduire des erreurs, et surtout adapter la conception du formulaire en fonction de l'objectif visé.

Les groupes d'items doivent être conçus en fonction des caractéristiques de la tâche : les champs doivent être regroupés dans des catégories basées sur la nature sémantique des éléments. En fonction des choix du concepteur, les groupes seront conçus:

- selon la séquence d'utilisation logique

- selon la fréquence d'utilisation

- et / ou selon l'importance relative

Les champs dont la probabilité de remplissage est moindre, qui sont optionnels, pourront être groupés dans une catégorie finale du dialogue, si cette règle ne contredit pas une autre règle de groupement plus importante (exemple : la séquence d'utilisation logique).

2.3. Alignement des champs et des libellés

Pour optimiser l'organisation visuelle d'un formulaire, on doit aussi se préoccuper des rapports libellés / champs. Il s'agit de jouer sur l'alignement pour que l'apparence du formulaire aide la lecture.

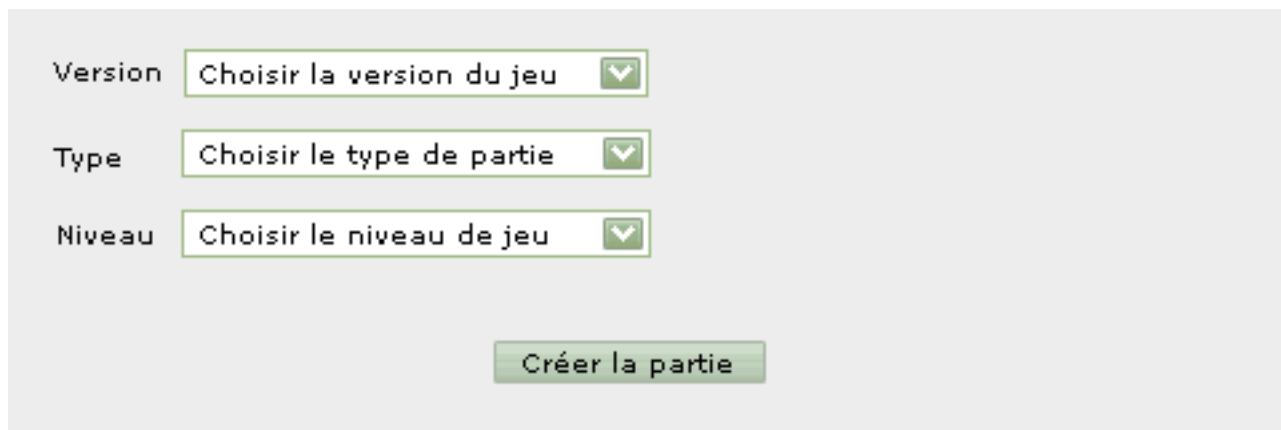
En règle générale, il est appréciable que tous les champs soient alignés. On essaie plutôt de jouer sur l'alignement des libellés aux champs. Ceci permet que l'organisation visuelle du formulaire conserve une certaine cohérence et soit plus facile à appréhender.

Les avantages d'un alignement à gauche des libellés sont essentiellement liés à la facilité de lecture : il est beaucoup plus facile de lire une liste de libellés alignés sur la gauche. Cependant, il n'est pas toujours recommandé d'aligner les libellés à gauche. C'est en fonction de la taille des libellés que l'on décidera d'un alignement à gauche ou à droite:

■ Alignement à gauche

Les libellés sont dits alignés à gauche lorsque qu'ils débutent tous au même endroit sur un repère horizontal.

On recommande d'aligner les libellés à gauche lorsque le nombre de caractères séparant le libellé le plus long de celui le plus court ne dépasse pas 6 caractères (cf. Mayhew dans les lectures complémentaires).



Version Choisir la version du jeu ▼

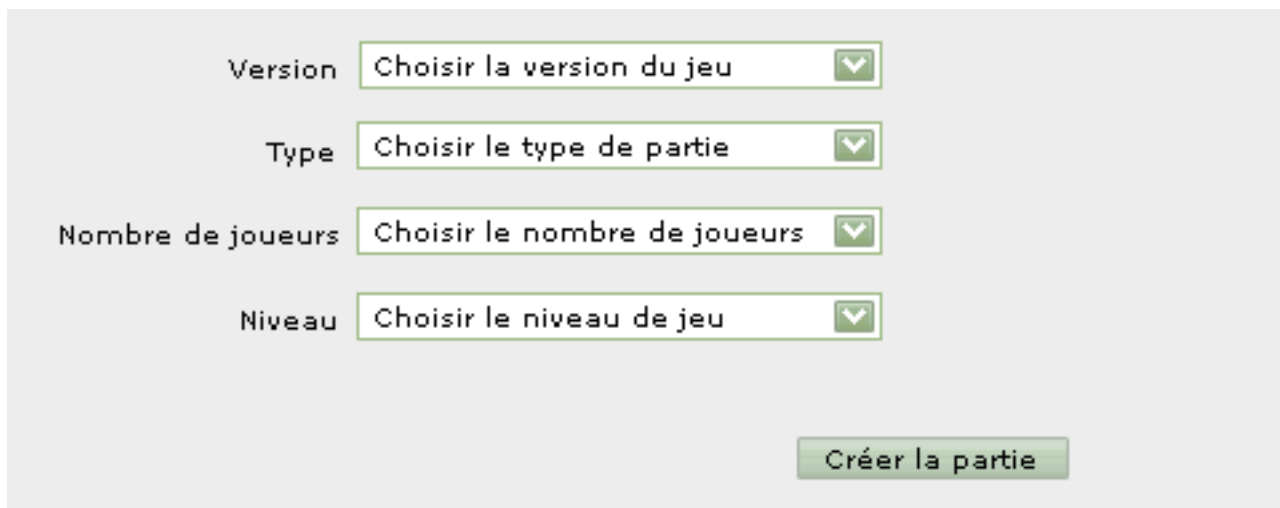
Type Choisir le type de partie ▼

Niveau Choisir le niveau de jeu ▼

Créer la partie

Exemple : Lorsque les libellés du formulaire ont une longueur comparable, il est plus adapté de les aligner à gauche. Ceci facilite en effet la lecture.

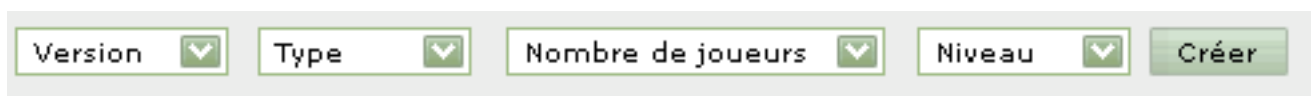
■ Alignement à droite



The image shows a form with four rows of labels and dropdown menus. The labels are 'Version', 'Type', 'Nombre de joueurs', and 'Niveau', all right-aligned. Each label is followed by a dropdown menu with the text 'Choisir la version du jeu', 'Choisir le type de partie', 'Choisir le nombre de joueurs', and 'Choisir le niveau de jeu' respectively. A 'Créer la partie' button is located at the bottom right of the form.

Exemple : les libellés de ce formulaire sont alignés à droite. Cela permet de conserver une distance libellé - champ raisonnable malgré la longueur du libellé “Nombre de joueurs”.

■ Libellé dans l’élément de formulaire



The image shows a form with four input fields and a 'Créer' button. The labels 'Version', 'Type', 'Nombre de joueurs', and 'Niveau' are placed inside the input fields, right-aligned. Each label is followed by a dropdown arrow. The 'Créer' button is on the right.

On peut placer le libellé dans le champ de formulaire afin de gagner de l’espace à l’écran. Ce type d’interface, souvent réservé à des applications ou des fonctionnalités en mode “expert”, permet d’accomplir rapidement et en prenant peu de place une action très fréquente.

2.4. Champs obligatoires versus champs optionnels

La lisibilité d’un formulaire passe aussi par une détection rapide de ce qui est “obligatoire” et de ce qui ne l’est pas. Il est donc essentiel de penser à spécifier les champs obligatoires.

Les champs obligatoires doivent être légendés. On utilise en général à côté du libellé du champ un élément coloré signifiant le caractère obligatoire (astérisque, point ou autre élément graphique).

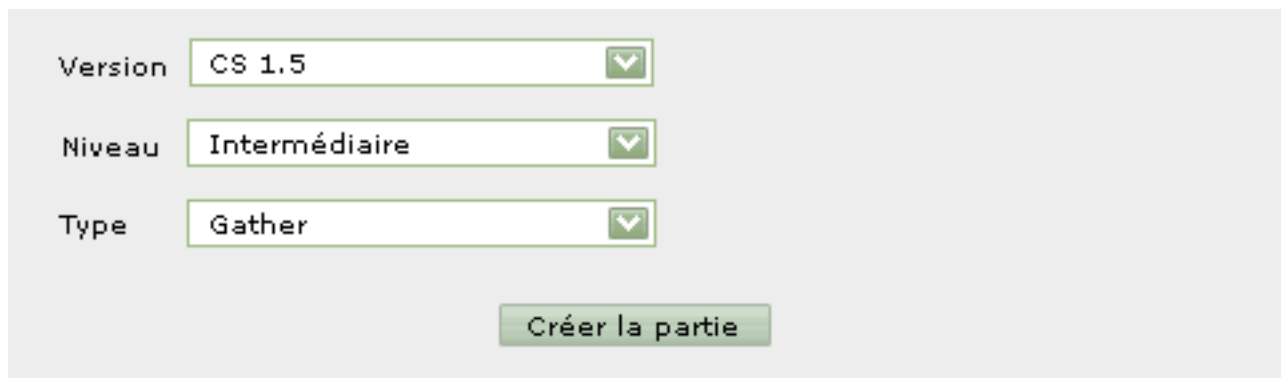
La légende de cet élément doit être placée avant ou après le formulaire. Il est plus adapté de la placer avant le formulaire, afin de respecter la logique de lecture de haut en bas. La légende peut être une phrase du type “Les informations marquées d’un * doivent être remplies / sont obligatoires”, “* indique un champ obligatoire”, etc.

3. Rapidité d’exécution

La question de la rapidité d’exécution est une préoccupation valable sur la plupart des formulaires. Il s’agit que l’utilisateur puisse remplir le formulaire le plus rapidement possible, sans toutefois être induit en erreur. Il existe plusieurs manières de limiter le temps de remplissage d’un formulaire:

■ Pré-sélection d'options

Exemple : On peut aider l'utilisateur à remplir le formulaire plus rapidement en fournissant des valeurs par défaut:



The image shows a form with three dropdown menus and a button. The first dropdown menu is labeled 'Version' and has 'CS 1.5' selected. The second dropdown menu is labeled 'Niveau' and has 'Intermédiaire' selected. The third dropdown menu is labeled 'Type' and has 'Gather' selected. Below the dropdown menus is a button labeled 'Créer la partie'.

On a spécifié pour chacun des champs une option par défaut, celle choisie la plus fréquemment. Cette pratique ne doit être appliquée que lorsque ce choix par défaut est prédominant, au moins pour 75% des utilisateurs.

■ Faciliter la sélection des options les plus fréquentes

Exemple : Pour faciliter la sélection des options les plus fréquentes, on peut découper les listes (par exemple les options d'un menu déroulant) en sous-groupes, en affichant dans le premier groupe les options les plus fréquentes.

Exemple : Enregistrement par le système d'exploitation des réponses déjà formulées. Il faut cependant fournir un moyen d'effacer ces entrées, et de refuser l'enregistrement des entrées.

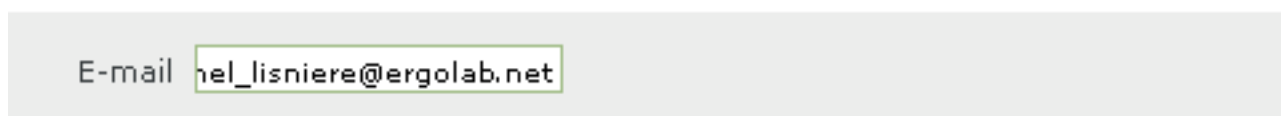
■ Faciliter le passage d'un champ à un autre

Exemple : On peut faciliter le passage au champ suivant pour les utilisateurs expérimentés en fournissant des raccourcis clavier. L'utilisation du clavier peut aussi permettre d'optimiser la sélection d'options (dans un menu déroulant par exemple).

Exemple : Passage automatique d'un champ au suivant lorsque l'on peut détecter que le premier est renseigné. Ce procédé ne doit être utilisé que pour les données de longueur fixe. En outre, il ne peut être appliqué que sur des interfaces utilisées très fréquemment et de façon répétée. Les utilisateurs doivent être experts et connaître parfaitement toutes les entrées du formulaire, leur ordre et leurs caractéristiques. Ce procédé peut poser problème sur les sites grand public: on ne connaît pas le niveau d'expertise des utilisateurs et ces utilisateurs ne viennent pas assez fréquemment pour "apprendre" le comportement de l'interface. De plus, il pose le problème de l'hétérogénéité du comportement en fonction des champs.

■ Faciliter la vérification des données entrées

Exemple : La longueur d'un champ (ligne ou texte) est un moyen de faciliter l'interaction des utilisateurs avec le formulaire, en limitant les actions à accomplir avec le clavier et la souris pour vérifier la validité des données entrées. En effet, lorsqu'un champ ligne est trop petit, les utilisateurs ne peuvent voir simultanément l'ensemble des données qu'ils ont renseignées. Ils sont donc obligés de se déplacer à l'intérieur du champ afin de vérifier qu'ils n'ont pas fait de fautes de frappe:



The image shows an email input field with the text 'hel_lisnierre@ergolab.net' entered. The field is highlighted with a green border.

Exemple : la longueur de ce champ E-mail est trop réduite pour que les utilisateurs puissent vérifier d'un seul coup d'oeil qu'ils ont bien renseigné leur adresse.

4. Sûreté d'exécution : la gestion des erreurs

4.1. Protection contre les erreurs

La gestion des erreurs dans un formulaire commence d'abord par ce que l'on appelle la protection contre les erreurs. Il s'agit de tout mettre en oeuvre pour ne pas induire l'utilisateur en erreur, et pour limiter les erreurs qu'il pourrait commettre.

La conception du formulaire est donc déjà un moyen de limiter en amont les erreurs potentielles. Mieux le formulaire est conçu, moins les utilisateurs risquent de commettre des erreurs.

Plus on fournit d'éléments de sélection (liste, menu déroulant, case à cocher, bouton radio), plus on diminue le risque d'erreur.

En effet, les éléments de formulaire qui présentent le plus de risque sont ceux qui acceptent la saisie de la part de l'utilisateur (champs ligne ou texte). On doit donc **prévoir les erreurs classiques**.

On peut par exemple, prévoir le fait que les utilisateurs attendant patiemment l'envoi du formulaire cliquent plusieurs fois sur le bouton de validation afin d'être certain que le formulaire soit envoyé. Si l'envoi du formulaire est long, on doit afficher une barre de progression. De plus, le système doit fournir un moyen informatique de corriger le double envoi.

On doit aussi limiter le risque de confusion entre un bouton "Annuler" et un bouton "Envoyer" en ne fournissant qu'un bouton à la fin du formulaire, celui de validation.

La **légende des champs** (par exemple indiquer que le mot de passe doit comprendre 6 caractères au minimum) permet d'indiquer à l'utilisateur les pièges potentiels du formulaire. Dans l'exemple précédent, la légende évite à l'utilisateur d'être contraint à passer par une étape supplémentaire s'il avait en tête un mot de passe inférieur à 6 caractères.

Protéger l'utilisateur contre les erreurs, c'est aussi lui donner la **possibilité de confirmer les actions importantes** (par exemple la validation finale d'une commande en ligne). Cela peut consister à récapituler les détails de ce que l'utilisateur va confirmer, et indiquer que le bouton de validation correspond à la validation finale, qu'il n'y a pas de possibilité de retour en arrière.

La **spécification des paramètres des éléments de formulaires** peut aussi permettre d'optimiser la protection contre les erreurs. Les valeurs des paramètres (exemple : taille visible du champ, nombre de caractères maximum) doivent être adaptées au type de données afin de faciliter le renseignement du formulaire par les utilisateurs.

La taille affichée d'un champ donne un indice aux utilisateurs sur quel type de donnée est attendu. Elle peut donc leur permettre d'aller plus rapidement dans la détection que l'information qu'ils pensaient renseigner n'est pas celle attendue, ou n'a pas le format attendu.

Par exemple, une saisie de dates présentée sous forme de trois champs différents (adaptés au format de données attendu) peut faire comprendre rapidement à l'utilisateur que le format de l'année attendu est sous forme de 4 chiffres, et non de 2. On évite ainsi qu'il commette une erreur et soit obligé de la corriger.

Enfin, la **validation du type de données renseigné** (caractère texte, chiffre, caractère spécial, etc.) peut permettre de détecter une erreur non remarquée par l'utilisateur, et évite que les données envoyées soient erronées.

4.2. Détection des erreurs

Il faut aussi concevoir le formulaire de façon “intelligente” au niveau du développement, afin de faciliter la détection des erreurs : par exemple, on doit checker si un code postal ne contient bien que des chiffres (si le formulaire est uniquement destiné à la France). Si l'utilisateur entre une donnée sous forme de caractères, on affiche un message d'erreur.

Il est important de ne pas “sur-valider” un formulaire, afin de rester flexible. Par exemple, pour un formulaire à visée internationale, la validation d'un champ de téléphone devrait accepter les caractères autres que les chiffres, tels que (,), +, etc.

La détection des champs doit se faire en une seule validation : on valide tous les champs en même temps et on affiche toutes les erreurs, au lieu de procéder de façon séquentielle (une erreur à la fois). Cela permet à l'utilisateur d'aller plus rapidement dans la correction de ses erreurs.

4.3. Affichage des messages d'erreur et aide à la correction

Deux problématiques différentes doivent être traitées dans l'affichage des messages d'erreurs :

- Le fond : qu'affiche-t-on dans les messages d'erreur?
- La forme : Comment affiche-t-on ces messages?

■ Contenu des messages d'erreur

On ne doit jamais retourner un formulaire sans en expliquer la raison. Un message général doit être fourni (qui explique “Vous avez fait une erreur”), mais aussi un / des messages spécifiques sous ou à côté des champs mal renseignés.

Ces messages doivent réellement expliquer l'erreur : il ne s'agit pas uniquement de dire “c'est faux”, mais plutôt “un code postal doit être composé uniquement de chiffres”, et fournir un exemple de renseignement correct si besoin.

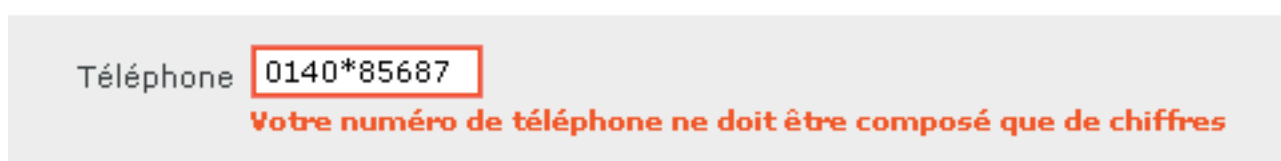
Suite à la validation, on devra veiller à ne pas effacer les entrées valides. De plus, pour faciliter la correction des erreurs, on n'effacera pas les entrées erronées.

■ Apparence des messages d'erreur

Afficher un message d'erreur dans la page et non dans une pop-up permet à l'utilisateur de ne pas avoir besoin de mémoriser la ou les erreurs : il peut revenir au message pour le relire plus attentivement si besoin.

Si le système technique vous contraint à afficher ce message dans une pop-up, vous pouvez fournir à l'utilisateur le moyen de corriger son ou ses erreurs directement dans cette pop-up.

Les messages d'erreur doivent être affichés de façon très distincte, dans une couleur de type “alerte” (rouge, orange). L'utilisateur ne doit pas se demander pourquoi la page s'est rechargée après validation (sans voir qu'un message d'erreur s'est affiché). Cette problématique semble peu importante, on pense que les utilisateurs verront forcément qu'un message d'erreur est apparu à l'écran. Cependant, des tests rapides peuvent montrer de façon criante l'importance du format des messages d'erreur. Vous pouvez à ce sujet consulter l'expérience de Viewit dans les lectures complémentaires.



Exemple d'affichage d'un message d'erreur : on affiche l'explication en rouge et en gras, sous le champ concerné. On accompagne ce message d'une mise en valeur du contour du champ mal renseigné.

The image shows a web form with the following fields and values:

- Nom:
- Prénom:
- Adresse:
- Code postal:
- Ville:
- Téléphone: (highlighted in orange)
- E-mail:

Below the telephone field, there is a red error message: **Votre numéro de téléphone ne doit être composé que de chiffres**. At the bottom of the form is a green button labeled **Valider**.

Exemple d’affichage d’un message d’erreur : on affiche l’explication en rouge et en gras, sous le champ concerné. On accompagne ce message d’une mise en valeur du fond du champ mal renseigné pour attirer l’attention de l’utilisateur sur ce champ en particulier.

Conclusion

Les problématiques de lisibilité, d’organisation visuelle et de gestion des erreurs sont les points centraux dans la spécification ergonomique d’un formulaire. La conception des formulaires est un domaine vaste et très contextuel. On peut dégager certains standards mais les choix d’interface sont souvent dictés par la spécificité des tâches.

Pour en savoir plus...

» Ressources en ligne

La visibilité des messages d’erreur, Viewit
(<http://www.hitit.com/viewit/exemple.asp#erreur>)

Des formulaires plus simples, tutoriel de Fred Cavazza
(http://www.fredcavazza.net/tutoriels/formulaire/SVF_intro.htm)

Usability and HTML Forms, Andrew Starling, 2001 (Web Developer’s Virtual Library)
(<http://wdvl.internet.com/Authoring/Design/Basics/form1.html>)

Enhancing form usability with instructions and validation, Michael Meadhra, 2004 (Builder.com)
(<http://builder.com.com/5100-6371-5200060.html>)

Usable Form Design, Your Usability Resource, 2004

Forms, usability, and the W3C DOM, Peter-Paul Koch, 2003 (Digital Web)
(http://digital-web.com/articles/forms_usability_and_the_w3c_dom/)

Why Users Don't Complain About Unusable Forms, Caroline Jarrett, 2001 (STC Usability SIG Newsletter)
(<http://www.stcsig.org/usability/newsletter/0101-forms.html>)

Forms - 508 Tutorial, Usability.gov
(http://www.usability.gov/web_508/tut-n.html)

Simple Tricks for More Usable Forms, Simon Willison, 2004 (Sitepoint)
(<http://www.sitepoint.com/print/1273>)

» Ressources externes

Mayhew (1992). Principles and Guidelines in Software User Interface Design, Prentice Hall. Chapitre 5 : Dialog Styles : Fill-in Forms.