**Sommaire**

I) Les points clefs  
 1) Les habitudes  
 2) Les normes  
 3) Les directives  
 4) Gestion des erreurs  
 5) Tests  
 6) Autres  
II) 10 Principes généraux de Jakob Nielsen  
 1) tenir informé l’utilisateur  
 2) cacher la complexité  
 3) utilisateur est libre, contrôle sa situation  
 4) respect des normes, directives, …  
 5) anticiper les erreurs  
 6) reconnaitre plutôt que mémoriser  
 7) flexibilité  
 8) esthétique  
 9) gérer les erreurs  
 10) documentation  
III) Les utilisateurs  
IV) Création d’une IHM  
 1) le card sorting  
 2) les wireframes et wireflow

**Interfaces homme-machine**

Les interfaces homme-machine qui est littéralement l’interface, souvent graphique qui est présentée à l’utilisateur lorsqu’il utilise l’application.

Créer une bonne interface nécessite de respecter ou de gérer correctement certains aspects. Le bon sens est primordial.

1. Les points clefs
2. Les habitudes

Lors de la conception d’une interface, on doit au maximum respecter les habitudes des utilisateurs ! (place du menu)

Le sens de lecture est de haut en bas, de gauche à droite.

Remarque : Vous écrivez une application pour les autres (normalement) et non pour vous !

1. Les normes

Il existe des normes ISO en termes d’ergonomie par exemple ISO-9241.

1. Les directives

Des systèmes d’exploitation, des éditeurs … donnent des directives qu’une application doit respecter en termes d’IHM si elle veut être publiée. Il est intéressant de les consulter même si cela ne nous concerne pas.

Ex (directives Gnome) :

* faire une application claire, sans trop de fonctionnalités
* afficher au fur et à mesure les actions complexes
* prévoir les possibles erreurs et les gérer proprement
* donner un beau nom d’application et une belle icone

1. Gestion des erreurs

Les erreurs sont un point important à gérer, il faut communiquer de façon claire à l’utilisateur.

Il n’y a pas de négation et on utilise la forme passive.

1. Tests

Il est important de faire tester son "IHM" par des utilisateurs variés pour vérifier la qualité de son IHM.

1. Autres

Le choix des polices est important et de leurs couleurs est importants.   
Le logiciel est facile à installer, désinstaller, n’interfère pas avec d’autres.

1. Principes généraux

10 principes généraux par Jakob Nielsen

1. toujours tenir informé l’utilisateur

Sur ce qu’il est en train de faire, de l’état du système (se demander ce qu’il est important qu’il sache), combien de temps il doit attendre…

On doit être sûr que ce que l’on a tenté de faire a marché ou non (clic).

Par exemple avec l’utilisation du fil d’Ariane, en mettant en avant la page actuelle, un bouton change au survol…

1. cacher la complexité à l’utilisateur et suivre les conventions

L’utilisateur se base sur ses expériences passés pour interagir avec l’interface, on doit faire en sorte de respecter ses habitudes.

Les menus de navigation doivent clairement afficher ce à quoi ils servent.

On peut donner des informations complexes dans la section « à propos ».

« Skeuomorph design » désigne le fait que les expériences digitales doivent correspondre aux expériences physiques (ex : pour un libre, on s’attend à pouvoir tourner les pages comme dans le monde physique)

1. utilisateur contrôle ce qu’il fait et est libre

On peut toujours revenir en arrière, par exemple avec « undo » et « redo », mettre fin à une action (bouton annuler).

L’utilisateur doit clairement voir qu’il a une sortie de secours.

1. voir 2, respect des normes, conventions, directives

Maintient interne : au sein d’une même famille de logiciels  
Maintient externe : de façon générale (ex : bouton panier sites marketing)

Cela ne concerne pas seulement des aspects visuels mais aussi le comportement derrière (on s’attend à ce que le bouton panier nous permette de valider nos achats…) : but facile à apprendre !

1. anticiper les erreurs

On empêche un problème d’arriver avant qu’il n’arrive, ce qui ne comprends pas seulement les problèmes techniques (l’utilisateur saisi une date au format y-m-d alors qu’on voulait d/m/y) mais également les accidents (utilisateur crée un compte avec un email mais ce n’était pas le bon).

On doit gérer ces cas, par exemple avec une confirmation avant de valider ou limiter les choix possibles de saisie.

On doit laisser à l’utilisateur le moins de chances possibles de faire (ou qu’il y ait) une erreur. (Utilisation de liste déroulantes/boutons de choix/sélection date dans calendrier de choix si possible pour éviter des erreurs de frappe).

1. reconnaitre plutôt que mémoriser

L’utilisateur ne doit pas avoir à retenir plein d’informations, mais va avoir des repères visuels pour se rappeler.

(ex : QCM plus facile pour utilisateur qu’une question ouverte)

On doit lui donner un maximum d’indices pour qu’il retrouve une information rapidement dans sa mémoire.

1. flexible (nouveau, anciens)

L’application doit être facile à prendre en main par des novices et permettre des raccourcis pour les anciens.

Les utilisateurs ont plusieurs options pour arriver à leurs fins.

1. esthétique

On ne doit montrer que les informations utiles (ne pas écrire un gros paragraphe pour 3 mots utiles, des images qui ne servent à rien…) qui complètent les informations nécessaires.

1. gérer les erreurs (+robustesse)

On prévient l’utilisateur lorsqu’une erreur arrive (mettre en rouge…), propose possiblement une solution (un lien qui règle problème).

1. Documentation

Elle peut être nécessaire pour des applications complexes même si, il est préférable de s’en passer (voir 6). Elle doit être facile à lire, l’utilisateur peut facilement trouver ce qu’il veut faire, liste des étapes à faire, utiliser des images…

1. Les utilisateurs

Tous les utilisateurs sont différents, pas seulement en termes de langues (pays), genre, âge, malvoyants, les daltoniens... mais également en terme de culture. Il n’existe pas d’utilisateur lambda et donc crée une application pour l’utilisateur lambda revient à créer une application pour personne.

On peut alors se rabattre sur des communautés et faire des vues adaptés pour chacune. (Version site français, en anglais, …)

On distingue également :

* si c’est le jour ou la nuit
* le matériel possédé par l’utilisateur (puissance du matériel, taille de l’écran, souris ? clavier ?...)
  + l’orientation de l’écran, les zones accessibles sur un téléphone, tablette sont différentes selon l’orientation
* sa connaissance des ordinateurs
* s’il sait lire

Remarque : les lecteurs lisent généralement en « F », c’est-à-dire que plus on descend et moins ils lisent.

1. Création d’une IHM

Voici une liste d’étapes possibles (Florent Madelaine) pour la réalisation d’une interface homme-machine.

* études des utilisateurs, on crée des catégories d’utilisateurs
* on fait des tests sur quelques utilisateurs (card sorting)
* on fait des diagrammes UML
* on fait une maquette papier et des wireframes (+wireflow)
* on fait des tests sur la maquette papier
  + regarde sur une wireframe ou l’utilisateur clique si on lui demande de chercher quelque chose
* on crée une maquette dynamique (html, css, js, boostrap)
* on fait encore des tests sur des utilisateurs

1. Le card sorting ou tri par cartes

On présente une liste des onglets du site web et on demande à l’utilisateur d’en faire des menus, ainsi que de donner des noms aux menus.

On peut ensuite montrer à d’autres utilisateurs des menus déjà faits et on leur demande de chercher un onglet. On regarde leur cheminement, s’ils trouvent directement, quels menus ils consultent…

On limite généralement à 7 le nombre de menu car c’est le nombre qu’une personne, en moyenne, retient.

Le site optimalworkshop propose de nombreux outils utiles pour les ergonomes dont le card sorting, mais également la possibilité de tester ses wireframes ou ses menus.

1. Wireframe et wireflow

Un wireframe est un écran de l’application à un instant t. Une page peut avoir plusieurs wireframe :

* page de connexion vide
* page de connexion avec erreur dans un champ
* page de connexion avec demande de confirmation
* ….

Le wireflow est le fait de lier toutes les wireframes entre elles.

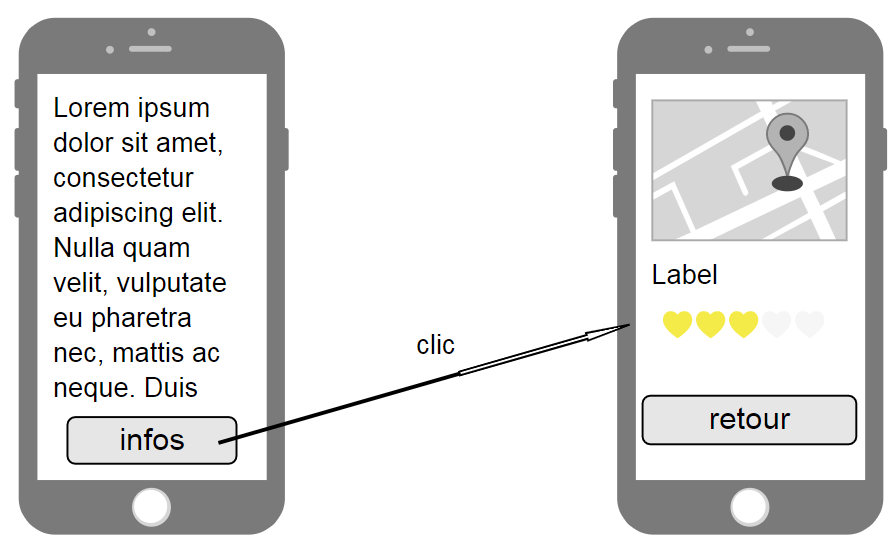
1. les wireframes

Les wireframe sont des dessins (qui peuvent être fait avec un logiciel comme moqups #pub) très minimalistes (simples) donc on donne juste une idée qu’ici il y aura un composant, ici un texte sans forcément écrire le texte.

1. le wireflow

Il s’agît d’un diagramme qui relie les wireframes, avec des flèches et possiblement une information sur comment/pourquoi on transite de cette wireframe A, à cette wireframe B.

Ils sont une version beaucoup moins complexe que les diagrammes d’état-transitions vus en UML.

Voici un exemple de wireflow avec deux wireframes en un passage en cliquant sur le bouton infos. (fait sur moqups en 2 min #pub2)

L’idéal est de tester son wireflow avec un ou plusieurs exemples (scénarios & synopsis d’UML).