Concepteur Développeur en Informatique

Développer des composants d'interface

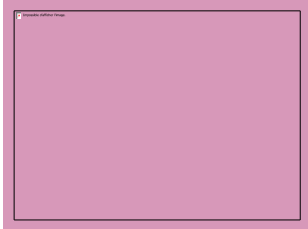
Conception Interfaces graphiques

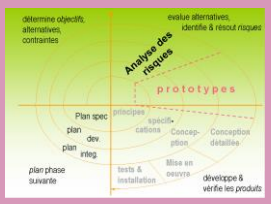
Accueil

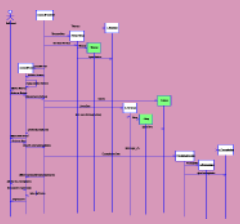
Apprentissage

PAE

Evaluation







Localisation : U03-E03-S02

SOMMAIRE

1. Introduction

Ce support aborde les principes de conception des interfaces Windows et les règles qui concourent à la réalisation d'interfaces homme machine respectant les règles de l'art.

2. Concevoir l'interface graphique

2.1. Programmation événementielle et procédurale

Une application Windows se distingue d'une application classique par le fait qu'elle se déroule dans un environnement graphique et que les interactions de l'utilisateur se font essentiellement à l'aide de la souris.
L'utilisateur n'est pas contraint de suivre la logique du programme.
Il fait ce qu'il veut dans l'ordre où il le souhaite et c'est le programme qui suit ses interactions.

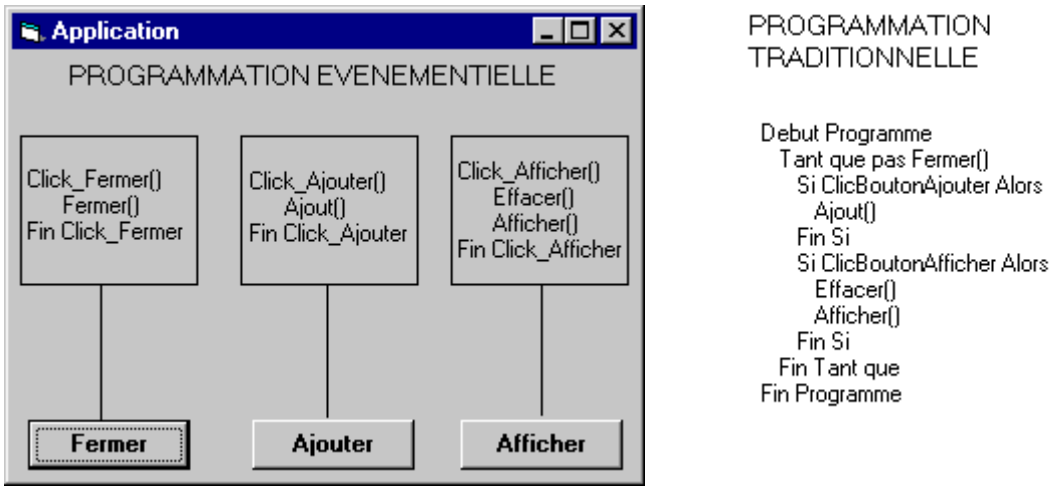


Figure 1 : Programmation événementielle et procédurale

Le code est exécuté lors de la survenance d'évènements qui peuvent être générés par le système, par programme ou par l'utilisateur.

L'environnement graphique Windows est une interface fonctionnant sur la logique événementielle. Son rôle consiste à scruter les actions de l'utilisateur et à redistribuer ces différents événements aux applications concernées.

De plus, le fonctionnement global de Windows est basé sur le couple objet/action. L'utilisateur désigne l'objet sur lequel il veut intervenir, puis choisit une ou des actions à effectuer. Ce principe est très proche de notre mode de fonctionnement. Il permet entre autre de définir des actions génériques applicables à différents objets.

Pour fonctionner ainsi, les applications doivent s'appuyer sur la logique événementielle et adhérer au principe objet/action. Or les applicatifs de gestion fonctionnent sur le modèle données/traitements. Associé à une interface graphique, les données sont considérées comme des objets (client, factures, …) et les traitements correspondent aux actions (ajouter, modifier, visualiser, supprimer, …).

2.2. LES ENJEUX DE L'IHM

La programmation événementielle est basée sur l'interaction entre l'utilisateur et l'application. Les différents objets de l'interface vont être manipulés indépendamment les uns des autres, en fonction de la tâche que l'utilisateur souhaite réaliser.

C'est l'utilisateur qui contrôle le déroulement de l'application.

De ce fait l'ergonomie d'une interface utilisateur est un des éléments primordiaux d'une l'application informatique.

2.2.1. Les conséquences d'une interface ratée

- Confusion : l'utilisateur est perdu dans l'application et ne sait plus où aller, ni que faire;
- Frustration : l'utilisateur est dans l'incapacité d'atteindre son but;
- Panique : l'utilisateur a peur d'avoir perdu des données ou détruit des fichiers;
- Stress : la charge de travail et/ou le nombre d'informations reçues sont trop importants pour l'utilisateur;
- Ennui et lassitude : l'utilisateur est fatigué d'avoir à répéter des étapes fastidieuses et non productives;
- Sous-utilisation : l'utilisateur refuse de se servir d'une application jugée trop complexe;
- Sur-utilisation : l'utilisateur doit exercer de multiples allers-retours entre les différentes fenêtres de l'application.

2.2.2. Les avantages d'une interface réussie

- Acceptation : l'utilisateur s'approprie l'application et s'en rend propriétaire;
- Efficacité : l'application est utilisée à bon escient et de manière efficace;
- Formation réduite : apprentissage minimal par l'utilisateur qui retrouve des automatismes liés à d'autres applications.

Le développeur ne peut pas se cantonner seulement aux outils professionnels, sans se préoccuper de l'environnement de travail habituel de l'utilisateur.
Il doit penser à la convergence entre l'environnement bureautique et l'environnement applicatif. Cette convergence ne peut être réalisée qu'en obtenant un haut niveau d'intégration des applications avec l'environnement qui les supporte
Cette intégration ne pourra se faire, si le développeur ignore les programmes manipulés habituellement par les utilisateurs.

Le défaut d'un développeur néophyte est de surcharger l'interface graphique de toutes sortes d'objets, rendant l'application impropre à une utilisation correcte. Il est donc nécessaire de s'attacher à un guide de style, véritable fil d'Ariane de la création d'applications graphiques.

Il faut, dans ces conditions, inclure une nouvelle étape dans la conception des projets, qui concernera l'interface. Cette étape se verra divisée en tâches successives, dont l'enchaînement ne sera ni rigide, ni irréversible.
Le développeur sera amené à modifier son interface en fonction des remarques de l'utilisateur et des aménagements successifs qu'il devra y apporter pour qu'elle respecte aux mieux les règles établies.

Il est donc préférable de créer une première version qui sera améliorée par la suite, en intégrant les remarques des utilisateurs, plutôt que de vouloir finaliser l'IHM dés le premier essai.

Le programmeur néophyte aura souvent tendance à être trop perfectionniste lors de la réalisation de l'interface ce qui induira des temps de développement plus longs. Pour minimiser les impacts sur les délais, le développeur sera tenté de minimiser les temps impartis à d'autres phases clés du développement, telles que les tests unitaires.

L'utilisateur doit absolument être associé à la création de l'interface. Il faut s'enquérir de ses besoins, de ses souhaits et de ses craintes.
Il est toutefois nécessaire de fixer un nombre maximum de cycles itératifs « conception- réalisation-présentation à l'utilisateur » nécessaires à la conception et à la mise au point des interfaces.
Eviter de dépasser 2 à 3 itérations.

La validation de l'interface ne devra se faire que lorsque les enchaînements seront réalisés.

2.3. LES PRINCIPES d'une interface Windows réussie

Il existe une norme intégrant un certain nombre de principes. Cette norme est nommée CUA (Common User Access – Interface Commune d'Accès). Plusieurs versions de cette norme cohabitent.

2.3.1. La reproduction du modèle conceptuel

Le développeur devra s'attacher à reproduire le modèle de fonctionnement de l'utilisateur dans son interface. La correspondance qui existera entre les deux mondes est la clé d'une application réussie (par exemple la calculatrice dans Windows).

Différentes techniques permettent d'être en phase avec le modèle conceptuel de l'utilisateur :

- Employer des métaphores : Réaliser des représentations imagées d'objets réels ou de situations sans toutefois tomber dans l'ambiguïté.
- Créer une ressemblance entre l'application et le domaine de l'utilisateur : Faire en sorte que l'utilisateur retrouve les automatismes qu'il avait acquis au cours de la réalisation manuelle de son travail à travers des méthodes simples et naturelles.
- Rendre l'interface cohérente : Respecter des standards de présentation, de comportement, de déroulement, de fonctionnalités.
- Rendre l'interface transparente : Masquer les mécanismes purement techniques à l'utilisateur. Sa seule préoccupation restant les tâches à accomplir et non la manière dont l'application les traite.

2.3.2. Laisser le contrôle de l'application à l'utilisateur

Ce doit toujours être l'utilisateur qui contrôle le déroulement de l'application et non l'inverse.

Le développeur doit faire en sorte que l'initiative du dialogue soit à l'avantage de l'utilisateur de l'application. Ce dernier doit pouvoir choisir la séquence des opérations pour effectuer une tâche.

L'utilisateur a le contrôle de l'application lorsqu'il est en mesure de basculer d'une activité à l'autre facilement, de changer d'avis en cours de saisie, d'arrêter une activité qu'il ne veut pas poursuivre, … Les techniques permettant ce contrôle sont les suivantes :

- Réaliser une interface indulgente : Pardonner les erreurs éventuelles de l'utilisateur, en lui permettant d'annuler les opérations qu'il ne souhaite pas valider. Demander une confirmation d'une opération lorsque celle-ci est irréversible.
- Faire une interface visuelle : Présenter des éléments visuels afin que l'utilisateur visualise les actions à réaliser, plutôt que de lui imposer de se souvenir du procédé à mettre en œuvre pour effectuer une opération.
- Donner un retour d'information immédiat : Transmettre une confirmation visuelle ou sonore et une information claire sur le résultat lors de certaines actions réalisées par l'utilisateur.

2.3.3. Cohérences des applications

C'est une règle fondamentale des applications s'appuyant sur l'interface graphique Windows. La cohérence consiste à homogénéiser l'apparence des différentes applications, la logique de réaction et le comportement. Elle favorise l'apprentissage des applications par capitalisation, les utilisateurs retrouvant leurs automatismes d'une application à l'autre. Quatre niveaux de cohérence doivent être respectés :

- Cohérence avec l'environnement graphique : Il y a, au sein même de Windows, une foule d'applications fournissant de précieux détails sur l'apparence, le comportement et la réactivité.
- Cohérence avec les autres applications : Il faut utiliser les composants de base de Windows et eux seuls, afin que toutes les applications aient une apparence et un comportement commun.
- Cohérence à l'intérieur de l'application : Il est impératif que l'application ait un caractère homogène. Tous les composants doivent se ressembler et réagir de façon similaire.
- Cohérence avec le monde réel de l'utilisateur : Il est nécessaire que l'application soit un miroir de la réalité. L'utilisateur doit y retrouver ses objets familiers.

2.3.4. Le dévoilement progressif

Le dévoilement progressif consiste à n'offrir à l'utilisateur que les actions qu'il peut réaliser en fonction de la situation dans laquelle il se trouve. Ce concept est indispensable à la réalisation d'une bonne interface graphique. Le risque encouru, si ce principe n'est pas utilisé, c'est de voir l'utilisateur noyé sous un flot d'informations, incapable de les mémoriser et de connaître celles qu'il doit utiliser.
Le développeur devra concevoir une barre de menu, où chaque élément représentera les grandes directions de l'application et les différents items des menus, les étapes incontournables.

2.3.5. Le retour d'information ou feedback

Le retour d'information est un principe énonçant qu'à chaque action réalisée par l'utilisateur est associé un rappel visuel simple indiquant que cette action s'est déroulée correctement. Ce principe revêt un caractère obligatoire car il est un élément essentiel de confort et de sécurité. Il répond au besoin de trouver après chaque action, une conséquence directe de cette dernière et à la nécessité d'en informer l'utilisateur.

La mise en œuvre permanente du retour d'information se traduit par des demandes de confirmation lors des actions importantes ou par l'indication visuelle et graphique de la progression d'un traitement. L'exemple le plus simple de retour visuel connu, réside dans le passage en inverse vidéo d'un texte sélectionné.

2.3.6. Les messages à destination de l'utilisateur

Les messages sont destinés aux utilisateurs. Ils doivent de ce fait être clairs, précis mais non répressifs. Tous les termes associés au langage informatique doivent être proscrits. Il existe quatre types de messages différents, symbolisés par quatre icônes :

- Le message d'information (symbole « i »), utilisé pour donner un simple compte rendu du résultat d'une action.
- Le message de confirmation (symbole « ? »), existant pour gérer le droit à l'erreur que doit posséder l'utilisateur et permettant de rendre l'interface indulgente.
- Le message d'avertissement (symbole « ! »), indiquant que l'application a détecté une anomalie n'ayant pas d'incidence immédiate sur son déroulement normal.
- Le message d'erreur (symbole « stop »), précisant comme précédemment que l'application a détecté une anomalie et que celle-ci doit être corrigée avant que soit poursuivi le traitement.

Il faut utiliser les messages d'avertissement et d'erreur avec parcimonie. L'interface de l'application doit prévenir plutôt que guérir. L'interface ne doit théoriquement pas fournir à l'utilisateur l'occasion de se tromper. Par exemple, dans le cas où les items d'un menu ne seraient pas utilisables, il faudra rendre leurs boutons inactifs.

Le texte des messages doit être le plus clair et le plus lisible possible. L'application doit proposer, si possible, une alternative à la situation et donner à l'utilisateur le moyen de résoudre le problème.

L'interface graphique d'une application doit permettre l'apprentissage par capitalisation. L'utilisateur doit apprendre de manière empirique en balayant les fenêtres, les enchaînements, toutes les possibilités offertes par l'application. Cette découverte ne doit avoir aucune conséquence sur les données, ce qui implique qu'à chacune des actions, devra être associée une procédure de retour en arrière.
Ce principe se retrouve au niveau des menus, que l'on peut dérouler sans avoir besoin de lancer le traitement correspondant, et également, dans les boîtes de dialogue où la logique validation/annulation est utilisée.

2.3.7. Multi-fenêtrage

Windows est une interface graphique multi-fenêtres. C'est à dire que l'écran est fractionné en plusieurs parties où s'exécutent les applications. Ces fenêtres peuvent se chevaucher, se recouvrir mutuellement ou être disposées côte à côte permettant ainsi de partager l'espace de travail.

2.3.8. Multi-contextes

Windows permet de travailler sur plusieurs applications à la fois. L'environnement multi-contextes est la partie visible du fonctionnement multi-tâches de Windows.
L'utilisation du multi-contextes présente par contre des désagréments pour le développeur. Chaque fenêtre et ainsi chaque application doit être clairement identifiée afin que l'utilisateur puisse aisément s'y retrouver.

De plus, plusieurs occurrences d'une même application peuvent fonctionner simultanément, chacune d'entre elles devant être repérée. Cela implique que le programmeur devra réfléchir à son application en tenant compte de l'environnement qui la supporte et devra réfléchir à la cohérence entre cette application et l'environnement. Dans ces conditions deux remarques s'imposent :

- L'application devra pouvoir communiquer avec l'environnement ;
- L'application ne devra pas monopoliser les ressources communes du système.

2.3.9. Notion de modalité

C'est une des notions les plus importantes de l'interface graphique. Elle décrit un mode de fonctionnement particulier. En effet dans un contexte modal, l'application reprend le contrôle. Elle attend une réponse de l'utilisateur et toutes les autres actions sont momentanément suspendues.
Par contre l'utilisateur peut accéder aux autres applications de Windows.
Pour simplifier, l'utilisation d'une fenêtre modale est nécessaire, dés lors que l'utilisateur doit terminer une action avant d'en commencer une autre.

Il est souhaitable d'éviter les situations modales au sein d'une interface graphique. Ce mode de fonctionnement allant à l'encontre même du principe de l'environnement graphique qui place l'utilisateur en position de contrôle de l'application. Malgré tout, dans certains cas, l'utilisation de ce contexte est inévitable :

- Les dialogues de continuation, demandant à l'utilisateur de fournir une information afin de continuer l'action en cours.
- Les actions orientées outils intervenant dans des applications où la désignation des objets provoquent une action.
- Les dialogues transactionnels, les plus fréquemment utilisés, attendant de la part de l'utilisateur la saisie d'informations.

Les situations modales sont inévitables de part la recherche de cohérence avec Windows qui les utilisent pour les interventions de l'utilisateur et pour respecter le contexte transactionnel, qui doit lors de traitement sur les données, gérer la concurrence d'accès et forcer l'utilisateur à répondre au principe validation/annulation.

2.3.10. Taille des applications

La taille des applications est un des aspects critiques de la performance globale d'un développement. Les applications doivent se limiter à un domaine fonctionnel précis, clairement délimité. Il est impératif de contenir la dimension d'un programme, selon les deux axes suivants :

- La largeur, c'est à dire le nombre de menus présents sur la barre de menus de la fenêtre principale. Le chiffre sept est déjà considéré comme étant trop important.
- La profondeur, représentant les enchaînements des dialogues entre eux. En dessus de trois enchaînements, l'application est considérée comme trop profonde.
- Un moyen simple de contenir la taille des applications est de limiter l'objectif du projet à un seul domaine fonctionnel. Tout ce qui n'est pas du ressort direct de la quasi-totalité des utilisateurs devant se servir de l'application, doit être mis dans une autre application.

2.3.11. SDI

Le SDI (Single Document Interface) est un mode de fonctionnement autorisant le chargement d'un seul document à la fois. Pour travailler sur un nouveau document, l'application doit au préalable fermer celui actuellement ouvert. Le principe des applications SDI est exploité dans le Bloc-notes de Windows.

Le mode SDI est à proscrire aujourd'hui.

2.3.12. MDI

Le MDI (Multiple Document Interface) autorise l'ouverture de plusieurs documents à l'intérieur d'une même application. Il définit le comportement des fenêtres documents à l'intérieur d'une fenêtre mère ou fenêtre de l'application. Les meilleurs exemples de l'utilisation de fenêtres MDI se trouvent dans les outils bureautiques Word et Excel. L'intérêt du principe MDI est suffisamment manifeste pour l'utiliser également dans des applications de gestion. Il se justifie alors dans l'ouverture simultanée de plusieurs vues du système d'informations de l'entreprise.
D'un point de vue technique, il permet de limiter les accès au réseau en stockant temporairement sur le poste client les données.

2.4. Construire l'interface

A partir du moment où le développeur dispose de la liste des fonctions qui seront mises en œuvre dans son application, les étapes de la réalisation de l'interface graphique devront être les suivantes :

- Les normes ;
- La fenêtre principale ;
- Le menu et éventuellement les menus contextuels ;
- Les fenêtres filles ;
- Les dialogues

2.4.1. Définition des normes

Il est nécessaire de définir certaines normes avant de commencer la conception et la réalisation de l'interface utilisateur. Le développement doit être rigoureux et soigné, pour cela on définit deux niveaux de normes :
Les normes de développement se présentent sous la forme d'une nomenclature des objets, des variables, les règles de codage.
Les normes d'ergonomie regroupent le choix des touches de fonction à utiliser, les raccourcis clavier, les options de dialogues, le nombre d'items autorisés à l'intérieur d'une liste ou d'une liste déroulante, …

2.4.2. La fenêtre principale

La fenêtre principale est la fenêtre de l'application. Toute application sous Windows s'exécute dans une fenêtre spécifique. Elle constitue l'espace de travail de l'application. Si la fenêtre principale de l'application est iconisée, c'est toute l'application qui passe en arrière plan. Il n'y a qu'une fenêtre principale par application. Elle contient toujours un titre (le nom de l'application), les boutons système, réduction et agrandissement. Elle n'est jamais modale mais peut être basée soit sur le principe MDI, soit plus rarement sur le principe SDI. A l'ouverture de l'application, elle occupe la totalité de l'écran.

Cette fenêtre contient, le menu de l'application, qui doit être unique, abstraction faite des menus contextuels, éventuellement une ou plusieurs barres d'outils et le cas échéant une barre d'état se situant en bas de la fenêtre. L'espace de travail est situé entre la barre d'outils et la barre d'état, il doit être suffisamment important puisque c'est à cet endroit que vont s'exécuter les fenêtres de traitement.

2.4.3. Le menu d'accueil et les menus contextuels

Le menu d'accueil est le menu général de l'application. C'est sans nul doute l'élément le plus complexe à définir, car toute l'application va dépendre de lui en termes d'ergonomie. Il est le point d'entrée de toutes les fonctionnalités de l'application et le point de départ de sa cinématique. A chacun de ses items est associée une fenêtre fille, une boîte de dialogue ou une action directe. Si ces items sont mal définis, c'est toutes les fenêtres descendantes que le développeur risque d'être obligé de redéfinir.
Le menu est toujours accessible à partir des fenêtres filles de l'application, mais jamais à partir des boîtes de dialogue.
Si un menu est contextuel, cela implique qu'il peut changer en fonction d'une situation donnée. Lorsqu'une fenêtre fille est active, tous les items du menu qui ne sont pas utilisables à partir de cette fenêtre doivent être grisés ou masqués.
Les menus standards de Windows comportent au minimum trois menus, quatre dans les applications basées sur le principe MDI. Ces menus sont les suivants :

- Le menu « fichier », positionné à gauche de la barre de menus, possède au moins l'item « Quitter » permettant de quitter l'application. Ce menu est en règle générale le point d'entrée dans l'application.
- Le menu « Edition » est le deuxième de l'application. Il permet de gérer le presse-papiers grâce aux items « Couper », « Copier » et « Coller ». Toute application Windows doit pouvoir gérer le presse-papiers.
- Le menu « Aide », présenté également par un point d'interrogation (« ? »). C'est le dernier menu de l'application et il contient toujours l'item « A propos de … » qui ouvre sur le dialogue correspondant.
- Le menu « Fenêtre », présent uniquement dans le cas d'applications MDI, se place en avant dernière position, juste avant le menu « ? ». Il comprend obligatoirement les items « Cascade », « Mosaïque » et « Réorganiser les icônes », ainsi que le nom des fenêtres filles ouvertes à ce moment là.

2.4.4. La fenêtre fille

La fenêtre fille, également nommée fenêtre secondaire ou fenêtre document contient généralement une liste de résultats ou une fiche d'informations. Elle peut être "icônisée" à l'intérieur de l'espace de travail. Elle peut également être arrangée en mosaïque ou en cascade. L'aspect général de cette fenêtre est identique à celui de la fenêtre principale de l'application, à ceci près qu'elle ne porte pas de menu, de barre d'outils et de barre d'état. Elle ne doit pas être employée pour effectuer des mises à jour sur des données, puisqu'elle ne dispose pas de boutons de commande.
Les contrôles pouvant lui être associés sont ceux permettant de filtrer les informations (liste déroulante, bouton radio, zone de texte, …).

2.4.5. La boîte de dialogue

C'est une fenêtre particulière, utilisée pour dialoguer avec l'utilisateur. Elle ne contient ni menu, ni barre d'outils, ni barre d'état, pas plus que d'ascenseur horizontal ou vertical. La boîte de dialogue n'est pas redimensionnable. Elle ne peut être iconisée et ne possède pas les boutons « Réduction » et « Agrandissement ».
La boîte de dialogue est indépendante de l'espace de travail. Elle est utilisée lorsque l'utilisateur doit réaliser une action.

De ce fait, elle possède les boutons de commande « OK » et « Annuler » permettant respectivement de valider ou d'annuler la procédure en cours.

2.4.6. Interdépendance

La profondeur de l'application ne doit pas excéder trois niveaux d'imbrication. Au-delà de cette limite, l'utilisateur risque de trouver l'application trop complexe. Dans le cas de dépendances fortes entre les boîtes de dialogue, ou d'une notion centrale dans l'application, il faut gérer l'information principale sur laquelle sont réalisées les opérations.
Le menu contextuel permet, lui, de traiter les actions.
Par exemple, si la donnée principale est le « Client », il sera souhaitable de choisir la valeur de cette donnée et ensuite de traiter des données qui lui sont rattachées comme les « Commandes », les « Factures », …

2.4.7. Fiche maître/esclave

Cette technique est basée sur la présentation dans la même fiche du référentiel et des données s'y rattachant. C'est une solution intéressante lorsque le volume de données à traiter est faible.
La partie maîtresse de la fiche affiche la liste des lignes de la table alors que la partie esclave présente l'ensemble des données d'une ligne de cette table. Cette seconde partie est renseignée automatiquement lors d'une action sur la liste de la partie maîtresse. En général, six boutons de commande sont présents sur ces fiches (voir schéma ci-après).

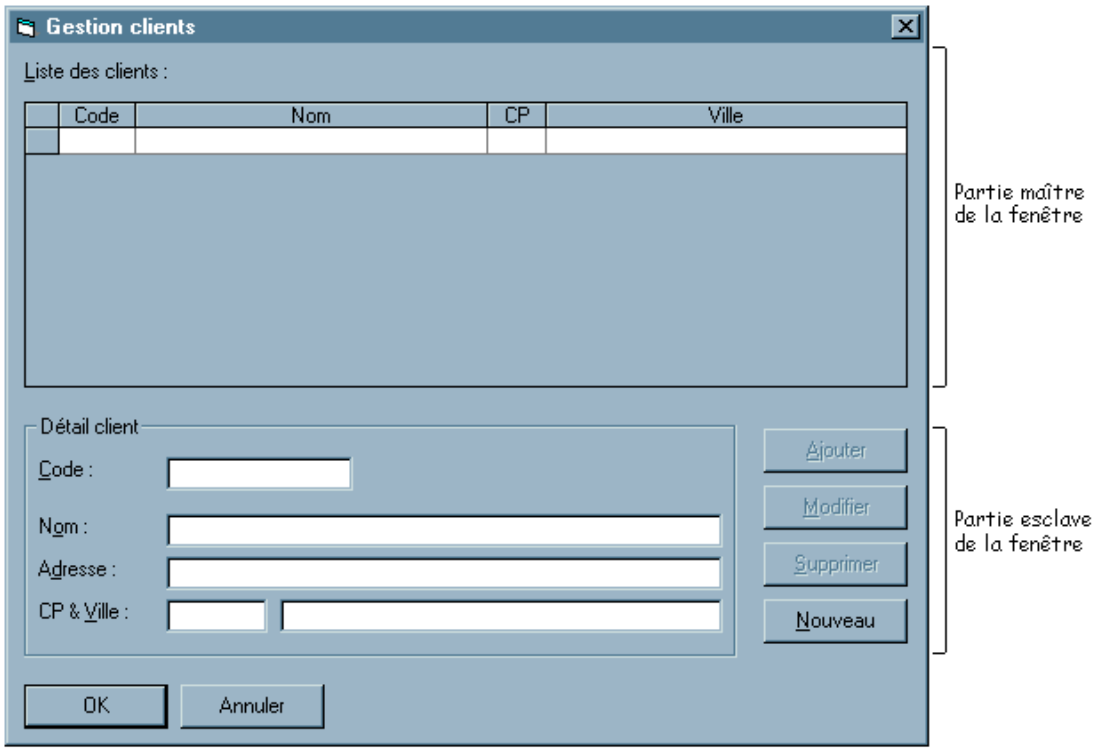


Figure 2 : Fenêtre Maître / Esclave

2.4.8. Fenêtre secondaire et dialogue

Cette technique sera préférable à la précédente dans le cas où le volume des lignes d'une table serait important.

La fenêtre secondaire contiendra alors une liste des lignes de la table et le cas échéant des critères de recherche. Les actions de recherche, d'ajout, de modification et de suppression sont accessibles à partir du menu contextuel, associé à la fenêtre. L'ajout et la modification sont réalisés à travers une boîte de dialogue. Chacune des actions est automatiquement répercutée dans la base de données.

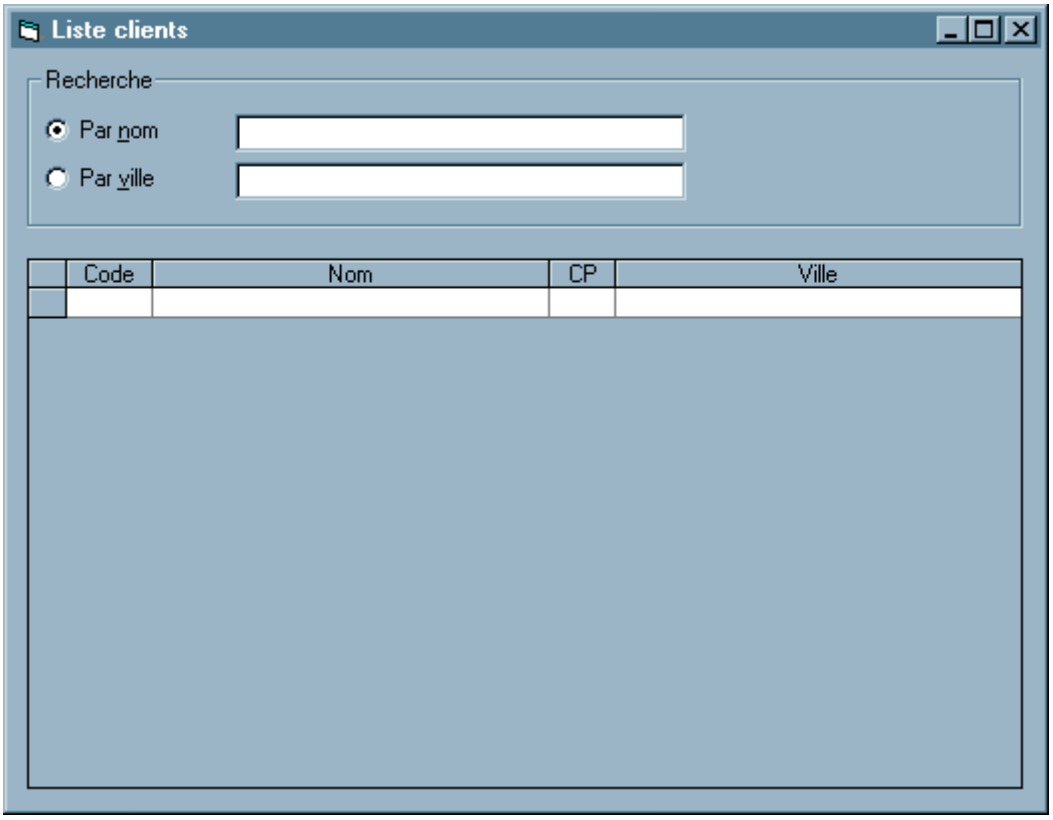


Figure 3 : Fenêtre secondaire

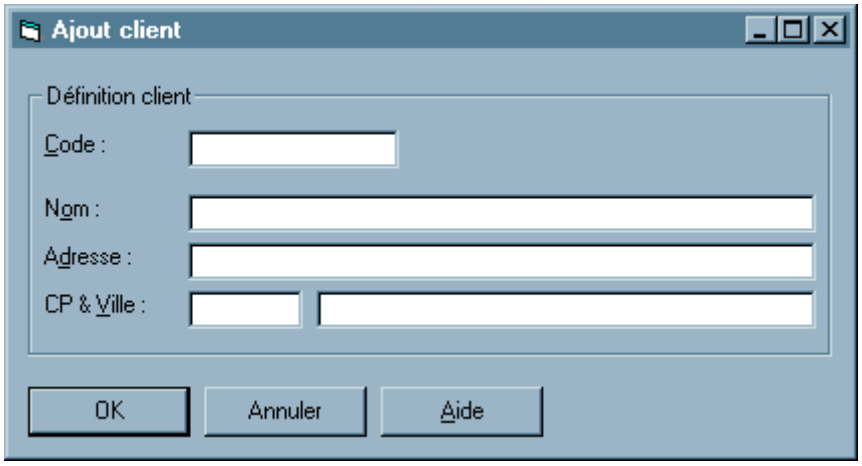


Figure 4 : Boite de dialogue