

Annuaire interactif en Mathematica

Conception d'une version simplifiée d'un annuaire interactif dans Mathematica

S. Klein (simon.klein@ens2m.org)

L. Lemattre (louis.lemattre@ens2m.org)

M. Munier (maxime.munier@ens2m.org)

Résumé : A l'aide des variables dynamiques, on propose de réaliser en Mathematica un annuaire interactif, accompagné de quelques applications à titre d'illustration et de test.

Mots clés : annuaire interactif, variable dynamique, liste

Abstract : Using dynamic variables, we propose to create an interactive directory in Mathematica, accompanied by some applications for illustration and testing purposes.

Keywords : interactive directory, dynamic variable, list

■ Introduction

Nous développons une application d'annuaire interactif en utilisant Mathematica, avec une interface conviviale grâce à DynamicModule et InputField. L'objectif est de simplifier la gestion des contacts avec des fonctionnalités telles que la validation, la modification et la suppression. Le programme permet également l'exportation/importation au format CSV. Ce rapport détaille le processus de développement, mettant en avant les choix de conception, les fonctionnalités clés, et les avantages pratiques du projet.

■ Étude préliminaire

■ Structure générale

Dans notre approche initiale pour stocker les données, nous envisagions la création de trois variables dynamiques distinctes (nom, prénom et numéro de téléphone). Cependant, après une réévaluation, nous avons opté pour une solution plus efficace en consolidant ces informations au sein d'une seule variable dynamique, sous la forme d'une liste de listes. Pour réaliser cette structure, nous avons fait usage de la fonction DynamicModule. Cette structure offre une flexibilité significative, notamment lors des opérations d'ajout ou de suppression d'entrées.

Au commencement de notre programme, nous définissons la liste de listes "info", qui pourra être sujette à des modifications ultérieures en raison d'opérations telles que l'ajout ou la suppression d'entrées. Pour accéder aux éléments spécifiques, tels que par exemple le nom, le prénom et le numéro de téléphone d'une personne, nous utilisons des expressions de type Dynamic, par exemple Dynamic[info[[1, 1]]] pour le nom.

```
DynamicModule[{{info = {{"Name_test", "FirstName_test", "Phone_test"}}, 
Grid[{ {"Nom :" , Dynamic[info[[1,1]]]}, {"Prénom :" , Dynamic[info[[1,2]]]}, {"Téléphone :" , Dynamic[info[[1,3]]]}]}]
```

Par ailleurs, nous avons pris la décision de concevoir notre programme de manière à ce qu'il s'ouvre dans une fenêtre distincte. Pour cela, nous faisons appel à la fonction CreateDialog. Cette approche vise à améliorer l'expérience utilisateur en offrant une interface dédiée et interactive pour la gestion des informations de l'annuaire, tout en restant intégrée au flux de travail global de Mathematica. La mise en œuvre d'une grille dynamique, garantit un affichage en temps réel des données. Cette fonctionnalité permet à l'utilisateur de visualiser instantanément les modifications apportées à l'annuaire.

```
(* [...]*)
CreateDialog[Column[{Grid[{
(* [...]*)
Dynamic[Grid[Prepend[info, {"Nom", "Prénom", "Téléphone"}], Frame→All]]}], 
WindowTitle→"Annuaire Interactif"]}]
```

Pour la version finale du programme, nous avons décidé de ne pas directement afficher la grille. Cette étape est mentionnée au cours du rapport, dans la partie récapitulatif du programme.

■ Ergonomie & Fonctionnalités

L'attention portée à l'ergonomie de notre application a été une considération primordiale afin de garantir une expérience utilisateur fluide et une interaction intuitive avec les données de l'annuaire. Initialement, notre réflexion s'est orientée vers l'utilisation de Manipulate, mais après réflexion, nous avons opté pour l'intégration de la construction Panel qui permet une meilleure flexibilité et un meilleur niveau de contrôle afin d'améliorer l'esthétique générale de l'interface utilisateur. Le choix stratégique d'éléments tels que Panel, Column et Row a permis de structurer l'interface de manière ordonnée. Panel, en particulier, crée une zone clairement délimitée, tandis que Column et Row sont employés de manière à organiser les éléments de façon verticale et horizontale, respectivement. Cette disposition contribue à une interface utilisateur bien agencée, favorisant la clarté et la facilité d'utilisation.

L'utilisation de InputField facilite grandement la saisie et la modification des informations de l'annuaire. Ces champs interactifs sont dynamiquement liés à la liste "info", assurant ainsi une cohérence entre les données entrées par l'utilisateur et celles affichées à l'écran. Cette approche renforce l'interaction utilisateur-application en garantissant une mise à jour en temps réel des données.

```
(* [...]*)
CreateDialog[Panel[Column[{Grid[{
{"Nom :", Dynamic[InputField[info[[1, 1]], String]]}],
 {"Prénom :", Dynamic[InputField[info[[1, 2]], String]]}],
 {"Téléphone :", Dynamic[InputField[info[[1, 3]], String]]}]}], 
(* [...]*)]
```

Le système de navigation de l'annuaire interactif est orchestré par une variable notée 'n'. Cette variable représente l'indice de l'entité actuellement affichée dans l'annuaire. Lorsque l'utilisateur veut changer de personne dans l'annuaire, la valeur de 'n' est ajustée en conséquence. Par ailleurs, des boutons interactifs ont été judicieusement incorporés pour offrir à l'utilisateur une navigation aisée entre les différentes entrées de l'annuaire. Ces boutons sont organisés au sein d'une barre de boutons structurée (ButtonBar) et comprennent des symboles intuitifs, tels que "«" (First), "<" (Previous), ">" (Next), et "»" (Last). Cette conception vise à faciliter la navigation séquentielle dans la liste des contacts, rendant ainsi l'expérience utilisateur à la fois fluide et intuitive. Les infobulles (Tooltips) ont été intégrées pour offrir des indications visuelles contextuelles lors du survol des boutons. Par exemple, le survol du bouton "«" affiche une infobulle indiquant le texte "First", offrant ainsi une indication immédiate sur la fonction associée à ce bouton. Cette approche renforce la facilité d'utilisation de l'application en guidant efficacement l'utilisateur.

```
DynamicModule[{"info" = {{ "", "", ""}}, n = 1},
(* [...]*)
Row[{ButtonBar[{Tooltip["«", "First"] → (n = 1),
Tooltip["<", "Previous"] → (n = Max[n - 1, 1]),
Tooltip[">", "Next"] → (n = Min[n + 1, Length[info]]),
Tooltip["»", "Last"] → (n = Length[info])}]}], 
(* [...]*)]
```

En outre, le système d'ajout et de suppression dans l'annuaire interactif est lié à la variable 'n'. Lorsqu'un utilisateur ajoute une nouvelle entrée, la liste 'info' est étendue, et 'n' est ajusté pour pointer vers cette nouvelle entrée. De même, lorsqu'une personne est supprimée, la liste 'info' est réduite, et 'n' est ajusté pour rester cohérent avec la

position actuelle dans la liste. On décide pour cela d'utiliser la fonctionnalité ActionMenu qui constitue un ajout significatif en offrant un accès rapide à des fonctionnalités avancées de l'application par l'intermédiaire de menu déroulant. Cette fonctionnalité étend les capacités pratiques de l'annuaire interactif, permettant à l'utilisateur d'accomplir des actions plus avancées tout en maintenant une interface utilisateur accessible sans surcharger l'interface de boutons.

```
(* [...]*)
ActionMenu[
  "Edit",
  {"Add" :> (info = Insert[info, {"", "", ""}, n + 1]; n = n + 1),
   "Delete" :>
     If[n == 1,
      PopupWindow[Text, "Erreur taille minimale atteinte"], (info = Delete[info, n]; n =
(* [...]*)]
```

On intègre aussi un menu “File” qui permet d'effectuer des opérations clés liées à la gestion des données. Ce menu, réalisé à l'aide de l'élément interactif ActionMenu, offre aux utilisateurs des fonctionnalités essentielles telles que l'exportation et l'importation de données, ainsi que la possibilité de quitter l'application. On utilise la fonction SystemDialogInput pour obtenir une interface de parcours de dossier et on utilise la fonction Export, pour exporter notre liste de liste info en une “Table” en utilisant comme séparateurs “;” pour correspondre au format CSV. On utilise aussi la fonction Import pour importer un fichier CSV dans notre annuaire.

```
(* [...]*)
ActionMenu[
  "File",
  {"Export CSV" :> Export[SystemDialogInput["FileSave", "data.csv"], info, "Table", "FieldSeparator" -> ";"],
   "Import CSV" :> (info = Import[SystemDialogInput["FileOpen"], "Table", "FieldSeparator" -> ";"]),
   "Exit" :> DialogReturn[]},
  Method -> "Queued"],
(* [...]*)]
```

L'attention minutieuse portée à l'ergonomie de notre application ne se limite pas uniquement à la structure de l'interface, mais s'étend également à des fonctionnalités d'affichage qui améliorent l'expérience utilisateur. Nous avons intégré un bouton permettant de basculer entre le mode clair et le mode sombre. Ce bouton, affichant dynamiquement un symbole représentant le soleil ☀ ou la lune ☽, offre à l'utilisateur la possibilité de personnaliser l'apparence visuelle de l'interface. Lorsque le mode sombre est activé, la couleur de fond de l'interface est modifiée, créant ainsi un contraste visuel qui peut être plus agréable dans des conditions de faible luminosité. Cette fonction est activée dynamiquement à l'aide de la fonction Dynamic, permettant une modification instantanée de l'apparence en réponse à l'action de l'utilisateur. Cet ajout offre un niveau supplémentaire de personnalisation, permettant à l'utilisateur de choisir l'environnement visuel qui lui convient le mieux. Lorsque le bouton est pressé, la variable darkMode, qui représente l'état du mode, est inversée (passant de vrai à faux ou inversement). En fonction de cet état, les options de style de la fenêtre d'évaluation et des cellules de l'interface sont ajustées pour refléter le choix du mode. Dans le cas du mode sombre, les options de fond de la fenêtre et des cellules sont définies sur un niveau de gris plus bas (GrayLevel[0.2]) pour créer un thème sombre. De plus, la taille et la couleur de la police de caractères sont ajustées pour améliorer la lisibilité dans un environnement sombre (police blanche quand la variable darkMode est à True). Ces ajustements sont spécifiés dans la section TextStyle.

```

DynamicModule[{info = {{ "", "", ""}}, n = 1, darkMode = False},
(*[...]*)
Button[Dynamic@If[darkMode, "○", "🌙"],
darkMode = Not[darkMode];
SetOptions[#, 
Background → If[darkMode, GrayLevel[0.2], White],
BaseStyle → If[darkMode, {White, 14}, {}]] & /@
Cells[EvaluationNotebook[],
CellStyle → {"Input", "Output", "Text"}];
SetOptions[EvaluationNotebook[], 
Background → If[darkMode, GrayLevel[0.2], White]]]],
Grid[{
{Dynamic[
Style["Nom : ", FontColor → If[darkMode, White, Black], Bold]],
InputField[Dynamic[info[[n, 1]]], String, ImageSize → {150, 25}]],
(*Idem pour prénom et nom *)
(*[...]*)
}
]

```

■ Récapitulatif du programme

En fusionnant les divers éléments composant la création de notre annuaire interactif, le programme résultant s'ouvre sous la forme d'une fenêtre simulant une interface interactive. Cette dernière est dotée de menus déroulants, de boutons d'action, de champs interactifs pour la saisie d'informations, et offre un affichage en temps réel sous forme de grille, présentant les différents contacts répertoriés dans l'annuaire. On propose le programme au complet dans l'annexe 1.

Dans la version finale du programme, nous avons pris la décision de ne pas directement mettre la grille d'affichage dans la première fenêtre mais nous avons décidé de rajouter un bouton “Preview” dans le menu déroulant “File” qui permet d'ouvrir une nouvelle fenêtre dans laquelle se trouve la grille d'affichage. Nous en profitons également pour rajouter une barre de défilement vertical permettant de gérer une quantité importante de données en utilisant “Scrollbars”.

```

DynamicModule[{info = {{ "", "", ""}}, n = 1, darkMode = False, previewWindow},
(*[...]*)
{ActionMenu["File",
(*[...]*)
"Preview" →> (previewWindow =
CreateDialog[
Dynamic@
Pane[Grid[
Prepend[info, {"Nom", "Prénom", "Téléphone"}],
Frame → All, Background → White,
ItemSize → {Automatic, 2},
Alignment → {{Left, Center}}],
Scrollbars → {False, True},
ImageSize → {Automatic, 150}],
WindowTitle → "Preview",
Background →
Dynamic@If[darkMode, GrayLevel[0.2], White]]),

```

Cependant, au cours du développement du projet, il est devenu évident que son potentiel dépasse celui d'un simple annuaire. En effet, en permettant l'ajout et la suppression de champs divers, au-delà des seuls nom, prénom et téléphone, le système peut évoluer vers une base de données interactive offrant une plus grande flexibilité dans la

manipulation des données. Dans l' annexe 2, nous présentons la version finale du projet qui inclut des fonctionnalités pour ajouter et supprimer des champs . La méthode employée pour intégrer ces fonctionnalités est détaillée dans la suite du rapport .

On introduit une variable “fieldTitles” pour gérer les titres des champs. Dans le premier programme, les titres des champs étaient codés en dur, tandis que dans le deuxième, ils sont dynamiques et modifiables. En outre, on ajoute aussi les variables dynamiques “newFieldTitle” et “removeFieldTitle” afin de permettre à l'utilisateur de modifier la structure de la base de données en ajoutant ou supprimant des champs.

```
DynamicModule[{{info = {{ "", "", ""}}, n = 1, darkMode = False,
  fieldTitles = {"Nom", "Prénom", "Téléphone"}, previewWindow,
  newFieldTitle = "", removeFieldTitle = ""}},
```

On modifie également le menu déroulant “Edit” pour permettre à l'utilisateur d'ajouter ou de supprimer des champs. Pour l'opération d'ajout, un champ de saisie “InputField” apparaît pour que l'utilisateur puisse entrer le nom du nouveau champ “newFieldTitle”. Après la saisie, un bouton “Confirm” permet à l'utilisateur de valider son choix. Si le nom du champ n'est pas vide “StringLength[newFieldTitle] > 0”, le programme ajoute ce nouveau titre de champ à la liste des titres de champs “fieldTitles”. Le nouveau champ est ajouté à chaque enregistrement existant dans la base de données “info”, initialement vide pour chaque enregistrement. Pour l'opération de suppression, une boîte de dialogue s'affiche avec un menu déroulant “PopupMenu” listant les champs existants “fieldTitles”. L'utilisateur choisit le champ à supprimer. Après la sélection, l'utilisateur confirme son choix en cliquant sur “OK”. Si le champ sélectionné fait partie des titres de champs existants “MemberQ[fieldTitles, removeFieldTitle]”, le programme trouve son index, le retire de la liste “fieldTitles” et supprime la colonne correspondante de chaque enregistrement dans info.

```
ActionMenu["Edit",
  (*[*]*)
  "Add Field" :> (newFieldTitle =
    InputField[Dynamic[newFieldTitle], String,
    FieldHint :> "Enter the name of the new field"];
    CreateDialog[{newFieldTitle,
      Button["Confirm",
        If[StringLength[newFieldTitle] > 0,
          AppendTo[fieldTitles, newFieldTitle];
          info = Map[Append[#, ""] &, info];
          DialogReturn[]]], WindowTitle :> "Add Field"]),
    "Remove Field" :> (removeFieldTitle =
      DialogInput[
        Column[{PopupMenu[Dynamic[selectedField], fieldTitles],
          Button["OK", DialogReturn[selectedField]],
          Button["Cancel", DialogReturn[None]]}]];
      If[MemberQ[fieldTitles, removeFieldTitle],
        removeIndex =
          Position[fieldTitles, removeFieldTitle, 1, 1][[1, 1]];
        fieldTitles = DeleteCases[fieldTitles, removeFieldTitle];
        info = Map[Delete[#, removeIndex] &, info];]),
      Method :> "Queued"],
    (*[*]*)
```

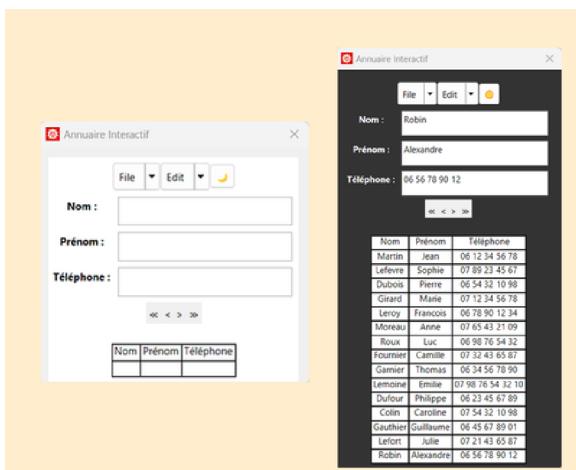
On doit également modifier l'affichage. On parcourt tous les titres de champs “fieldTitles” et on crée une ligne pour chaque champ.

```
(* [...]*)
Dynamic[
  Grid[
    Table[
      With[{i = i},
        {Dynamic[
          Style[fieldTitles[i]],
          FontColor → If[darkMode, White, Black], Bold]],
        InputField[Dynamic[info[n, i]], String,
          ImageSize → {150, 25}]]}, {i, Length[fieldTitles]}],
      ItemSize → {Automatic, 2}]]],
(* [...]*)
```

■ Applications

■ Annuaire interactif

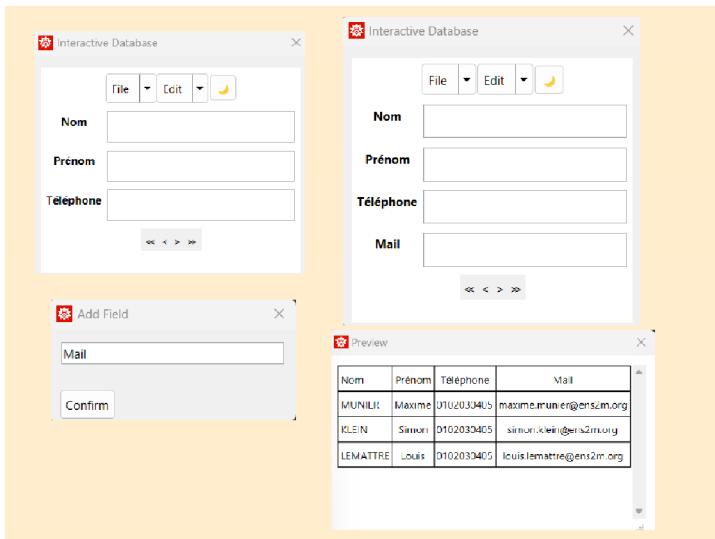
On suppose disposer d'un fichier regroupant 15 contacts sous le format nom;prénom;téléphone. On exécute d'abord le programme, on obtient la fenêtre à gauche. On décide de passer en mode sombre, et on utilise le menu déroulant afin d'importer notre fichier CSV. On obtient alors la fenêtre à droite



On peut alors naviguer dans l'annuaire et modifier les informations souhaitées directement dans les champs de saisie. On peut aussi supprimer certains contacts ou en ajouter d'autres en utilisant le menu déroulant "Edit". Une fois les modifications effectuées, on peut exporter les données au format CSV.

■ Base de données interactive

On ajoute un champ "Mail" via le bouton "Add Field" dans le menu déroulant "Edit". On rentre certaines données puis on accède à la fenêtre d'aperçu via le bouton "Preview" du menu déroulant "File". On peut également supprimer certains champs et on peut toujours exporter nos données au format CSV comme dans le premier programme.



■ Discussion et perspectives

■ Discussion

Le choix d'utiliser une liste de listes comme variable dynamique dans ce programme présente plusieurs avantages. Premièrement, il permet de structurer les données de manière organisée, où chaque sous-liste représente une entrée de l'annuaire avec les informations nécessaires (nom, prénom, téléphone). Cela facilite la manipulation des données dans le code, notamment pour l'ajout, la suppression et la modification des entrées. Les variables dynamiques en Mathematica, notées avec le mot-clé 'Dynamic', sont essentielles pour créer des interfaces utilisateur interactives. Elles permettent de lier des expressions dynamiques aux objets graphiques, comme les champs d'entrée ('InputField') et les boutons, assurant une mise à jour automatique de l'interface en réponse aux changements des variables. Le code utilise des 'InputField' dynamiques liés à des éléments spécifiques de la liste de listes 'info'. Cette approche est plus concise et extensible que d'utiliser trois 'InputField' distincts pour chaque champ. La variable 'info' est mise à jour automatiquement lorsqu'un champ d'entrée est modifié, grâce à l'utilisation de la fonction 'Dynamic'.

L'utilisation des associations pourrait également être une option envisageable pour la réalisation de ce projet. Elles permettent de stocker des associations clé-valeur, ce qui pourrait être utile pour représenter les données de manière plus explicite. Cependant, l'utilisation de listes de listes est souvent plus simple et suffisante pour des structures de données de taille modérée comme celles utilisées dans cet exemple. Si l'on souhaitait utiliser les associations, nous pourrions attribuer une clé unique à chaque entrée de l'annuaire et stocker les informations associées. Cela pourrait rendre certaines opérations plus rapides, mais cela pourrait également ajouter de la complexité au code.

En conclusion, le choix entre listes de listes et associations dépend des besoins spécifiques du programme. Les listes de listes offrent une approche simple et intuitive, tandis que les associations pourraient être préférables dans des cas plus complexes nécessitant une récupération efficace d'informations à partir de clés spécifiques.

■ Perspectives

Lorsque le numéro de téléphone dans le fichier CSV est de la forme 0786967354 et non pas 07 86 96 73 54, alors l'importation ne fonctionne pas correctement car dans le premier cas Mathematica considère que le type de donné est Number alors que dans le deuxième cas, il s'agit d'un String.

```
(* Si l'on modifie *)
InputField[Dynamic[info[[n, 3]]], String,
(* Par *)
InputField[Dynamic[info[[n, 3]]], Number,
```

Alors les 0 seront effacés car considérés comme inutiles. On décide donc de garder String et de faire attention au format dans le CSV. Il faudrait intégrer des mécanismes de validation pour s'assurer que les données saisies sont dans un format valide. Par exemple, vérifier la validité des numéros de téléphone ou imposer des contraintes sur les champs.

Lorsqu'on importe un fichier CSV avec des champs qui ne sont pas créés dans notre liste de champs interactive, les champs d'édition “*InputField*” n'apparaissent pas. Cependant les données sont bien importées car elles sont visibles depuis le menu “*Preview*”. Il faudrait modifier la partie du programme sur l'importation pour lire les différents champs du fichier CSV et les créer en conséquences dans notre liste “*FieldTitles*”.

On pourrait aussi ajouter des fonctionnalités pour filtrer ou trier les données de l'annuaire en fonction de critères spécifiques, comme l'ordre alphabétique des noms ou la recherche par prénom.

■ Conclusion

L'expérience de développement de ce micro-projet a mis en évidence l'efficacité et la puissance du langage Mathematica. La construction *DynamicModule* a joué un rôle essentiel en permettant la création d'une interface réactive et intuitive, facilitant ainsi la gestion interactive de l'annuaire. Cette immersion dans la programmation fonctionnelle propre à Mathematica a enrichi nos compétences informatiques, en particulier dans la manipulation de variables dynamiques et la gestion d'éléments graphiques interactifs. Au-delà de l'aspect technique, la conception de l'interface utilisateur a souligné l'importance de l'ergonomie dans le développement du projet. Mathematica, en tant que langage, offre de larges possibilités, ce qui en fait un outil polyvalent pour des projets couvrant un large spectre, de la modélisation mathématique avancée à la création d'applications interactives. L'intégration de Mathematica dans ce projet a ainsi produit une application fonctionnelle, renforçant simultanément la prise de conscience quant à l'importance d'adopter une approche globale du développement logiciel, et à la fois performances techniques et expérience utilisateur.

■ Bibliographie

- Wolfram Language Documentation. (s. d.). <https://reference.wolfram.com/language/>
- Trott Michael : The Mathematica GuideBook for Programming, Springer Science & Business Media, 2004
- Aide-mémoire pour Mathematica. (s. d.). <https://www.universite-paris-saclay.fr/>

■ Annexe

■ Annexe 1 : programme annuaire interactif

```

DynamicModule[{info = {{", ", ""}}, n = 1, darkMode = False},
CreateDialog[
Panel[
Column[
{Row[
{ActionMenu["File",
 {"Export CSV" :>
 Export[SystemDialogInput["FileSave", "data.csv"], info,
 "Table", "FieldSeparators" -> ";"],
 "Import CSV" :> (info = Import[SystemDialogInput["FileOpen"],
 "Table", "FieldSeparators" -> ";"]; n = Length[info]),
 "Exit" :> DialogReturn[]], Method -> "Queued"],
ActionMenu["Edit",
 {"Add" :> (info = Insert[info, {"", "", ""}, n + 1];
n = n + 1),
 "Delete" :>
If[n == 1,
 PopupWindow[Text, "Erreur taille minimale atteinte"], (info = Delete[info, n];
n = Min[n, Length[info]])]],
Button[Dynamic@If[darkMode, "○", "⊖"],
darkMode = Not[darkMode];
SetOptions[#, 
Background -> If[darkMode, GrayLevel[0.2], White],
BaseStyle -> If[darkMode, {White, 14}, {}]] & /@
Cells[EvaluationNotebook[], 
CellStyle -> {"Input", "Output", "Text"}];
SetOptions[EvaluationNotebook[],
Background -> If[darkMode, GrayLevel[0.2], White]]], 
Grid[{{Dynamic[
Style["Nom : ", FontColor -> If[darkMode, White, Black], Bold]],
InputField[Dynamic[info[[n, 1]]], String, ImageSize -> {150, 25}]},
{Dynamic[
Style["Prénom : ", FontColor -> If[darkMode, White, Black], Bold]],
InputField[Dynamic[info[[n, 2]]], String, ImageSize -> {150, 25}]},
{Dynamic[
Style["Téléphone : ", FontColor -> If[darkMode, White, Black], Bold]],
InputField[Dynamic[info[[n, 3]]], String, ImageSize -> {150, 25}]}]},
Row[{ButtonBar[
{Tooltip["<", "First"] :> (n = 1),
Tooltip["<", "Previous"] :> (n = Max[n - 1, 1]),
Tooltip[">", "Next"] :> (n = Min[n + 1, Length[info]]),
Tooltip[">", "Last"] :> (n = Length[info])}]}, Spacer[1],
Dynamic[
Grid[Prepend[info, {"Nom", "Prénom", "Téléphone"}],
Frame -> All, Background -> White]], Alignment -> Center],
Background -> Dynamic@If[darkMode, GrayLevel[0.2], White]],
WindowTitle -> "Annuaire Interactif"]}]

```

■ Annexe 2 : programme base de données interactive

```

DynamicModule[{info = {{ "", "", ""}}, n = 1, darkMode = False,
  fieldTitles = {"Nom", "Prénom", "Téléphone"}, previewWindow,
  newFieldTitle = "", removeFieldTitle = ""},
 CreateDialog[
  Pane[Panel[
   Column[
    {Row[
     {ActionMenu["File",
      {"Export CSV" :>
       Export[SystemDialogInput["FileSave", "data.csv"], info,
        "Table", "FieldSeparators" -> ";"],
       "Import CSV" :> (info =
          Import[SystemDialogInput["FileOpen"], "Table",
          "FieldSeparators" -> ";"];
          n = Length[info]),
       "Preview" :> (previewWindow =
          CreateDialog[
           Dynamic@Dynamic@
            Pane[Grid[Prepend[info, fieldTitles]], Frame -> All,
             Background -> White, ItemSize -> {Automatic, 2},
             Alignment -> {{Left, Center}}],
            Scrollbars -> {False, True},
            ImageSize -> {Automatic, 150}],
           WindowTitle -> "Preview",
           Background ->
            Dynamic@If[darkMode, GrayLevel[0.2], White]]),
       "Exit" :> DialogReturn[]]}, Method -> "Queued"],
     ActionMenu["Edit",
      {"Add Record" :> (info =
         Insert[info, Table["", {Length[fieldTitles]}], n + 1];
         n = n + 1),
       "Delete Record" :>
        If[n == 1,
         PopupWindow[Text,
          "Erreur taille minimale atteinte"], (info =
          Delete[info, n];
          n = Min[n, Length[info]])],
       "Add Field" :> (newFieldTitle =
          InputField[Dynamic[newFieldTitle], String,
          FieldHint -> "Enter the name of the new field"];
          CreateDialog[{newFieldTitle,
           Button["Confirm",
            If[StringLength[newFieldTitle] > 0,
             AppendTo[fieldTitles, newFieldTitle];
             info = Map[Append[#, ""] &, info];
             DialogReturn[]]]}, WindowTitle -> "Add Field"]),
       "Remove Field" :> (removeFieldTitle =
          DialogInput[
           Column[{PopupMenu[Dynamic[selectedField], fieldTitles],
            Button["OK", DialogReturn[selectedField]],
            Button["Cancel", DialogReturn[None]]}]];
          If[MemberQ[fieldTitles, removeFieldTitle],
           removeIndex =

```

```
Position[fieldTitles, removeFieldTitle, 1, 1][1, 1];
fieldTitles = DeleteCases[fieldTitles, removeFieldTitle];
info = Map[Delete[#, removeIndex] &, info]];
Method → "Queued"],
Button[Dynamic@If[darkMode, "○", "☽"],
darkMode = Not[darkMode];
SetOptions[#, 
    Background → If[darkMode, GrayLevel[0.2], White],
    BaseStyle → If[darkMode, {White, 14}, {}]] & /@ 
Cells[EvaluationNotebook[]],
CellStyle → {"Input", "Output", "Text"}];
SetOptions[EvaluationNotebook[],
Background → If[darkMode, GrayLevel[0.2], White]]}],
Dynamic[
Grid[
Table[
With[{i = i},
{Dynamic[
Style[fieldTitles[[i]],
FontColor → If[darkMode, White, Black], Bold]],
InputField[Dynamic[info[[n, i]]], String,
ImageSize → {150, 25}]], {i, Length[fieldTitles]}],
ItemSize → {Automatic, 2}}],
Row[{ButtonBar[
{Tooltip["<<", "First"] → (n = 1),
Tooltip["<", "Previous"] → (n = Max[n - 1, 1]),
Tooltip[">", "Next"] → (n = Min[n + 1, Length[info]]),
Tooltip[">>", "Last"] → (n = Length[info])}], Alignment → Center],
Spacer[1]}, Alignment → Center],
Background → Dynamic@If[darkMode, GrayLevel[0.2], White]],
{Automatic, Automatic}],
WindowTitle → "Interactive Database "]]
```