```c
static void setup_file_list_item(GtkSignalListItemFactory *factory,
                                 GtkListItem *list_item,
                                 FileListData *data)
{
    FileListItem *item = gtk_list_item_get_item(list_item);

    GtkWidget *box = gtk_box_new(GTK_ORIENTATION_HORIZONTAL, 12);
    gtk_list_item_set_child(list_item, box);

    GtkWidget *filename_label = gtk_label_new(item->filename);
    gtk_box_append(GTK_BOX(box), filename_label);

    GtkWidget *type_label = gtk_label_new(item->file_type);
    gtk_box_append(GTK_BOX(box), type_label);

    GtkWidget *delete_button = gtk_button_new();
    gtk_button_set_icon(GTK_BUTTON(delete_button), data->delete_icon);
    gtk_box_append(GTK_BOX(box), delete_button);

    g_signal_connect_swapped(delete_button, "clicked", G_CALLBACK(delete_file), item);
}
static void bind_file_list_item(GtkSignalListItemFactory *factory,
                                GtkListItem *list_item,
                                FileListData *data)
{
    FileListItem *item = gtk_list_item_get_item(list_item);

    GtkWidget *box = gtk_list_item_get_child(list_item);
    GtkWidget *filename_label = gtk_widget_get_first_child(box);
    GtkWidget *type_label = gtk_widget_get_next_sibling(filename_label);
    GtkWidget *delete_button = gtk_widget_get_next_sibling(type_label);

    gtk_label_set_label(GTK_LABEL(filename_label), item->filename);
    gtk_label_set_label(GTK_LABEL(type_label), item->file_type);
      //g_object_bind_property() un peu option
    g_object_bind_property(item, "filename", filename_label, "label", G_BINDING_SYNC_CREATE);
    g_object_bind_property(item, "file_type", type_label, "label", G_BINDING_SYNC_CREATE);
}

static FileListData *create_file_list_data(GListStore *store, GdkPaintable *delete_icon)
{
    FileListData *data = g_new(FileListData, 1);
    data->store = g_object_ref(store);
    data->delete_icon = g_object_ref(delete_icon);
    return data;
}
```

```c
static GtkListItemFactory *create_file_list_factory(GdkPaintable *delete_icon, GListStore *store)
{
    GtkListItemFactory *factory = gtk_signal_list_item_factory_new();
    FileListData *data = create_file_list_data(store, delete_icon);
    g_signal_connect(factory, "setup", G_CALLBACK(setup_file_list_item), data);
    g_signal_connect(factory, "bind", G_CALLBACK(bind_file_list_item), data);
    return factory;
}
static GtkWidget *create_file_list_view(GListModel *model, GdkPaintable *delete_icon)
{
    GtkWidget *list_view = gtk_list_view_new(model, create_file_list_factory(delete_icon, model));
    gtk_list_view_set_show_separators(GTK_LIST_VIEW(list_view), TRUE);
    return list_view;
}

static GListModel *create_file_list_model(GList *file_list, GdkPaintable *delete_icon)
{
    GListStore *store = g_list_store_new(FILE_LIST_ITEM_TYPE); // create a GListStore
    for (GList *l = file_list; l != NULL; l = l->next)
    {
        const gchar *filename = (const gchar *)l->data; // Glist contient juste le nom du fichier
        const gchar *file_type = __get_file_mime_type(filename);
        FileListItem *item = g_new(FileListItem, 1); // g_new genere un pointeur FileListItem
        item->filename = g_strdup(filename);
        item->file_type = g_strdup(file_type);
        item->delete_icon = g_object_ref(delete_icon);
        g_list_store_append(store, item);
    }
    return G_LIST_MODEL(store);
}
```

```c
typedef struct
{
    gchar *filename;
    gchar *file_type;
    GdkPaintable *delete_icon;
} FileListItem; // doit etre declare avant G_DEFINE_TYPE

typedef struct
{
    GListStore *store;
    GdkPaintable *delete_icon;
} FileListData;

static GList *listFile = NULL;

/* creation d'un GType */
#define FILE_LIST_ITEM_TYPE (file_list_item_get_type())
G_DECLARE_FINAL_TYPE(FileListItem, file_list_item, FILE, LIST_ITEM, GObject)
G_DEFINE_TYPE(FileListItem, file_list_item, G_TYPE_OBJECT)
```

```c
pSw = gtk_scrolled_window_new();
gtk_window_set_child(GTK_WINDOW(pWindowMain), pSw);
delete_icon = gtk_picture_new_for_filename("/image/delete2.png");
model = create_file_list_model(listFile, GDK_PAINTABLE(delete_icon));
pListView = create_file_list_view(model, GDK_PAINTABLE(delete_icon));
gtk_scrolled_window_set_child(GTK_SCROLLED_WINDOW(pSw), pListView);
```