|  |
| --- |
| ICalMerge |

Candidat : Leonardo Luna

Chef de projet : Jonathan Melly

Expert 1: Ernesto Montemayor

Expert 2: Serge Wenger

Table des matières

[1 Analyse préliminaire 3](#_Toc71699401)

[1.1 Introduction 3](#_Toc71699402)

[1.2 Objectifs 3](#_Toc71699403)

[1.3 Planification initiale 5](#_Toc71699404)

[2 Analyse / Conception 6](#_Toc71699405)

[2.1 Concept 6](#_Toc71699406)

[2.2 Stratégie de test 14](#_Toc71699407)

[2.3 Risques techniques 14](#_Toc71699408)

[2.4 Planification 14](#_Toc71699409)

[2.5 Dossier de conception 15](#_Toc71699410)

[3 Réalisation 15](#_Toc71699411)

[3.1 Dossier de réalisation 15](#_Toc71699412)

[3.2 Description des tests effectués 16](#_Toc71699413)

[3.3 Erreurs restantes 16](#_Toc71699414)

[3.4 Liste des documents fournis 16](#_Toc71699415)

[4 Conclusions 16](#_Toc71699416)

[5 Annexes 17](#_Toc71699417)

[5.1 Résumé du rapport du TPI / version succincte de la documentation 17](#_Toc71699418)

[5.2 Sources – Bibliographie 17](#_Toc71699419)

[5.3 Journal de travail 17](#_Toc71699420)

[5.4 Manuel d'Installation 17](#_Toc71699421)

[5.5 Manuel d'Utilisation 17](#_Toc71699422)

[5.6 Archives du projet 17](#_Toc71699423)

# Analyse préliminaire

## Introduction

Selon le Cahier des charges : ICalMerge est une solution pour fusionner plusieurs sources de calendrier au format ICAL vers un seul flux. Ceci est particulièrement utile lorsqu'on reçoit plusieurs calendriers pour une conférence par exemple et qu'on veut regrouper cela dans un seul fichier à importer…

Voici le croquis de l’application fourni avec le cahier des charges :



## Objectifs

1. **Maximum de 10 fichiers source**Il sera possible d’ajouter d’autres sources. Il pourra y en avoir un maximum de dix.
2. **Chargement de fichiers**

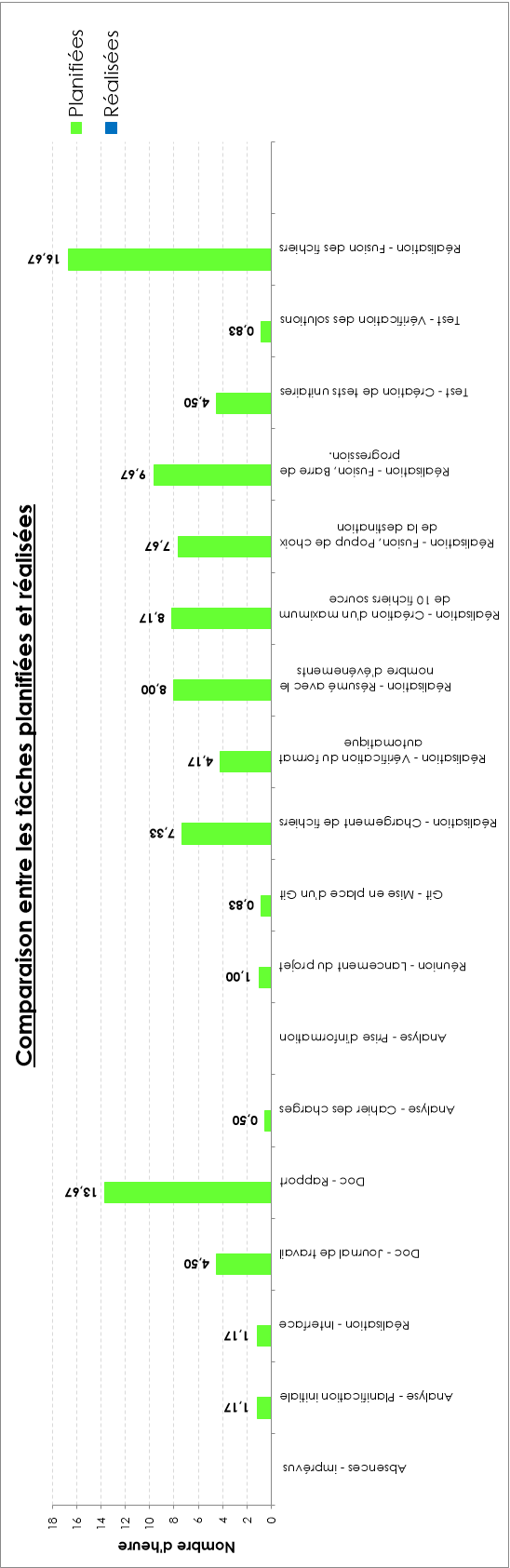
Il devra être possible d’importer deux chemin de fichiers source. Il sera possible d’ajouter les fichiers source par glisser/déposer, comme demandé dans les points techniques évalués du CDC.

1. **Vérification du format automatique**Lorsque l’utilisateur importera le chemin d’une source, une vérification du format du fichier devra se faire automatiquement.
2. **Résumé avec le nombre d’événements**Si l’importation d’un fichier s’est correctement déroulée, un label affichera le nombre d’événements que contient le fichier choisi. Voir image ci-dessous.



1. **Pop-up de fusion**Lorsque l’on clique sur le bouton « Fusionner », une pop-up s’ouvrira et permettra à l’utilisateur de choisir la destination du fichier résultant de la fusion.
2. **Barre de progression**Lorsque la fusion s’effectue, un barre de progression se charge en fonction de l’avancement de la fusion. Comme discuté avec M. Melly, le cent pourcent de la barre, correspondra au nombre total d’événements à fusionner.
3. **Avertissement fichier de destination**Lorsque la fusion se termine elle affichera un pop-up qui permettra d’exporter le fichier fusionné. Si le fichier de destination existe déjà, il faudra avertir l’utilisateur.
4. **Vérification de l’intégrité du fichier fusionné**Pour vérifier cela, le nombre d’événements créés devra correspondre au nombre d’événements à fusionner.
5. **Utilisation d’un système de versioning**Le projet devra être sauvegardé sur un programme de versioning
6. **Rubrique d’aide**Lorsque l’utilisateur pressera sur la touche F1 de son clavier, une page d’aide devra apparaître avec des instructions sur l’utilisation de l’application.
7. **Explication du format ICal**Une explication du format ICal sera requise dans le rapport de projet.
8. **Respect des normes de codage ETML**Le développeur devra suivre les normes de codages de l’ETML.

## Planification initiale



# Analyse / Conception

## Concept

1. **Maximum de 10 fichiers source**Pour commencer avec ce projet il serait bon de s’occuper de la partie graphique. Il faudrait qu’il soit possible d’ajouter jusqu’à 10 sources qui seront au début, purement visuelles. Celles-ci seront gérées par un objet nommé « SourceComponents ».  
     
   Un « SourceComponents » sera un objet qui contiendra les contrôles nécessaire pour que l’utilisateur puisse ajouter un fichier source.  
   Sur cette image il y’a deux lignes, donc deux fois cet objet :  
     
     
   Cela implique que chaque « SourceComponents » devra se charger de stocker le chemin d’un fichier source dans une variable de type string.  
      
   A chaque initialisation d’un de ces objet, il s’occupera d’afficher ses contrôles lui-même.  
     
   Pour savoir si nous avons atteint le nombre maximum de sources, le formulaire contiendra une liste de « Source Components ».  
     
   Pour dix objets dans cette liste, il y’aura dix lignes qui permettront d’ajouter une source. Donc, dix objets est égal à 10 sources. Pour ajouter une source il y’aura un bouton prévu cet effet :  
     
   Il serait dérangeant de pouvoir seulement en ajouter mais de ne pas en enlever. Pour cela un bouton de suppression sera prévu à cet effet.  
     
   Ce bouton impliquera qu’on retira le dernier « SourceComponents » de la liste et que l’on supprimera donc la dernière ligne ajoutée.

1. **Chargement de fichiers**Il y’aura deux façons possible d’importer un fichier à fusionner :

* Glisser/Déposer le fichier sur le textbox d’une source.
* Cliquer sur un bouton « parcourir » qui ouvrira une fenêtre permettant de choisir le fichier à importer.

1. **Vérification du format automatique**La classe « SourceComponents » contiendra une méthode qui se chargera de vérifier si le fichier indiqué est valide.  
     
   Cette méthode se lancera à chaque fois que :

* L’utilisateur modifie le chemin directement via le champ de texte
* Lorsque l’utilisateur importe un fichier source via le bouton « parcourir »
* Lorsque l’utilisateur utilise la fonctionnalité Glisser/déposer

Car toutes ces actions impliquent que l’utilisateur veut charger un fichier à fusionner.  
  
Si le chemin est valide et que le fichier est au format .ics, le champ de texte tu texbox sera rempli par le chemin du fichier. Dans le cas contraire, une message le label chargé de donner le nombre d’événement affichera « fichier invalide ». Cela empêchera l’utilisation du bouton de fusion.

1. **Résumé avec le nombre d’événements**Lorsque l’on importe un fichier .ics valide, label affichera le nombre d’événements qu’il contient. Cela s’apparentera à cela : OK : 8 événements.  
     
   Dans le cas où l’utilisateur essaierait d’importer un fichier non valide, le label affichera : KO : Invalide.  
     
   Voici une image de ces deux types de réponses que peuvent retourner ces labels, selon le cahier des charges :
2. **Pop-up de fusion**Lorsque toutes les fichiers sources sont valides, il sera possible de cliquer sur le bouton « Fusionner »  
   Il ne sera utilisable uniquement si l’utilisateur a inséré des fichiers sources valides. Dans le cas où un de ces fichiers aurait 0 événements, le fichier ne sera pas considéré comme valide et devra être remplacé ou l’objet « SourceComponents » devra être effacé. On pourrait très bien faire la fusion avec 0 événements, mais je souhaite écarter cette possibilité car :  
   - Cela ne servirait à rien de fusionner un fichier vide.  
   - Cela pourrait potentiellement générer des bugs et imprévus.

- L’utilisateur pourrait s’être trompé de calendrier et en avoir pris un vide.  
  
Lorsque tous les calendriers entrés ont été validés par le programme, le bouton de fusion permettra d’ouvrir une fenêtre d’exportation. L’utilisateur devra choisir l’endroit où il souhaite sauvegarder le fichier fusionné. Si l’utilisateur ferme cette fenêtre, rien ne se passe. Il n’y a pas de fusion. Dans le cas où il désigne un dossier où il souhaite mettre la fusion, une méthode de fusion de lance. Cette méthode parcourra chaque fichier et cherchera toutes les lignes commençant par « BEGIN :VEVENT ». Car cela implique que les prochaines lignes seront des propriétés liées à l’événement.

Dans cet exemple la méthode copiera tout de « BEGIN :VEVENT » à « END :VEVENT ». La dernière ligne impliquant la fin de l’événement.  
  
Entre ces deux lignes, il y’en a trois qui sont liées aux caractéristiques de la tâche.

* DSTART : Début de l’événement (date et heure)
* DTEND : Fin de l’événement (date et heure)
* SUMMARY : Titre de l’événement

Toutes ces lignes seront ajoutées à une liste de données. Cette liste contiendra tous les événements à fusionner.

Pour que le calendrier créé soit valide il nécessite deux additions.

Il faut que tous les événements soient entre deux lignes précises.

La première qui devra toujours être la première ligne du calendrier :

* BEGIN :VCALENDAR

La dernière définissant la fin du calendrier qui se doit d’être la dernière ligne du fichier :

* END :VCALENDAR

Voici un exemple de calendrier avec deux événements :

Cette exemple fonctionne car :

* Le fichier commence par « BEGIN :VCALENDAR »
* Les deux événements commencent par « BEGIN :VEVENT »
* La fin des deux événements est définie par « END :VEVENT »
* La dernière ligne annonce la fin du calendrier par « END :VCALENDAR »

**Le fichier fusionné aura le même schéma :**  
  
Début du calendrier

* Début événement 1
* Fin événement 1
* Début événement 2
* Fin événement 2

Fin du calendrier

1. **Barre de progression**

C:\Users\leoluna\AppData\Local\Microsoft\Windows\INetCache\Content.Word\19.ProgressBar.PNGVoici l’emplacement de la barre de progression selon le cahier des charges :  
  
J’ai choisi de garder cet emplacement car il est logique qu’elle s’applique pendant la fusion, donc après avoir interagit avec le bouton « Fusionner ».  
  
J’ai décidé que j’utiliserai l’outil Progress fourni avec le framwork .NET windows form de visual studio.

Elle se présente ainsi avec un taux de complétion de 50% :



**Pour l’utiliser il faut utiliser trois de ses variables de base :**

* « **Maximum** » définissant le 100% de la barre
* « **Minimum** » définissant le 0% de la barre
* « **Value** » définissant l’état actuel de la barre. S’il est égal ou plus petit que le minimum, la barre ne montrera aucune progression. Si la valeur est égale ou supérieure au maximum, la barre affichera une progression complète

Comme discuté avec le chef de projet, ces valeurs devraient être définies selon le nombre de tâches que l’on souhaite fusionner. Cela se manifeste sous cette forme :

* « **Maximum** » Définit le maximum de tâches à importer
* « **Minimum** » définit par le nombre 0. Car cela définit que l’on a importé 0 événements pour l’instant.
* « **Value** »définit par le nombre d’événements que l’on a déjà fusionné. Si l’on a fusionné douze événements, cette valeur sera égale à douze. Cette valeur sera actualisée à chaque importation d’un événement.

1. **Vérification de l’intégrité du fichier fusionné**

La vérification s’effectue à l’instant où tous les événements ont été fusionnés dans un seul fichier. Ce fichier devra être parcourru pour qu’on y compte le nombre d’événements final.   
  
Pour définir si la fusion a fonctionné, il faudra que le nombre d’événements contenu soit égal au nombre total d’événements de chaque calendrier source.

Voici un schéma représentant une fusion réussie :



Dans le cas où il n’y aurait pas d’égalité, un message erreur s’affichera à l’utilisateur. Il lui sera expliqué que les calendriers n’ont pas réussi à être fusionnés.

1. **Avertissement fichier de destination**Cette fonctionnalité s’applique lorsque l’utilisateur clique sur le bouton « Fusionner ». S’il choisit un emplacement où un fichier du même nom existe déjà, on affiche un avertissement.  
     
   Pour cela, nous allons utiliser un outil du framework .NET windows form de visual Studio. Cela s’appelle un « FolderBrowserDialog ».

Il permet d’ouvrir une fenêtre qui permettra à l’utilisateur de sélectionner un dossier. Cela sera utile pour définir où l’on voudrait stocker le calendrier (.ics) résultant de la fusion.

Voici un test que j’ai effectué pour pouvoir déterminer son utilisation.

J’ai créé un FolderBrowserDialog et j’ai laissé le nom par défaut « folderBroswerDialog1 ». J’ai utilisé la méthode ShowDialog() pour pouvoir afficher la fenêtre souhaitée.

Dans le cas d’une implémentation, le nom de cet objet serait apporprié à son utilisation. Cela n’est qu’un test pour but de comprendre comment utiliser cet objet.  
  
Voici la commande utilisée :

Voici la fenêtre que cela a ouvert :

Pour choisir le dossier un dossier, il faut cliquer sur celui souhaité et cliquer sur le bouton « OK ». Pour aller chercher un dossier qui se trouverait à l’intérieur d’un autre il faudra cliquer sur l’icône de flèche à côté du dossier parent.



Une fois que l’utiisateur a pu choisir le dossier qui va contenir la fusion de calendrier, il sera possible de retrouver son chemin. Il faut utiliser la variable « SelectedPath » qui retourne le chemin du dossier sous la forme d’un string.

L’utilisateur pourra choisir le nom du fichier fusionné. Pour cela, un nouveau formulaire s’ouvrira. Il contiendra un label «Nom du fichier » ainsi qu’un champ de texte qui lui permettra d’écrire le nom souhaité.  
Un bouton « OK » lui permettra d’envoyer sa réponse au programme.

**Une vérification s’effectuera.**   
  
**Si** le chemin entré précédemment est toujours valide et que le nom de fichier n’est pas déjà utilisé, le fichier (.ics) pourra être créé à l’endroit désigné.  
  
**Si** le nom est déjà utilisé à l’emplacement, un message s’affichera pour que l’utilisateur choisisse un autre nom.   
  
**Si** le nom est valide mais que l’utilisateur a supprimé une partie du chemin, il faudra qu’il entre un nouveau chemin.

À tout moment le processus de d’exportation pourra s’arrêter en fermant le formulaire ci-dessous ou en fermant la fenêtre du « FolderBrowserDialog » de la page précédente.  
  
Cela sera utilise si par exemple, l’utilisateur se rappelle qui souhaite ajouter un autre calendrier à fusionner. Cela impliquerait qu’il ne souhaitera poursuivre cette fusion.

1. **Utilisation d’un système de versioning**

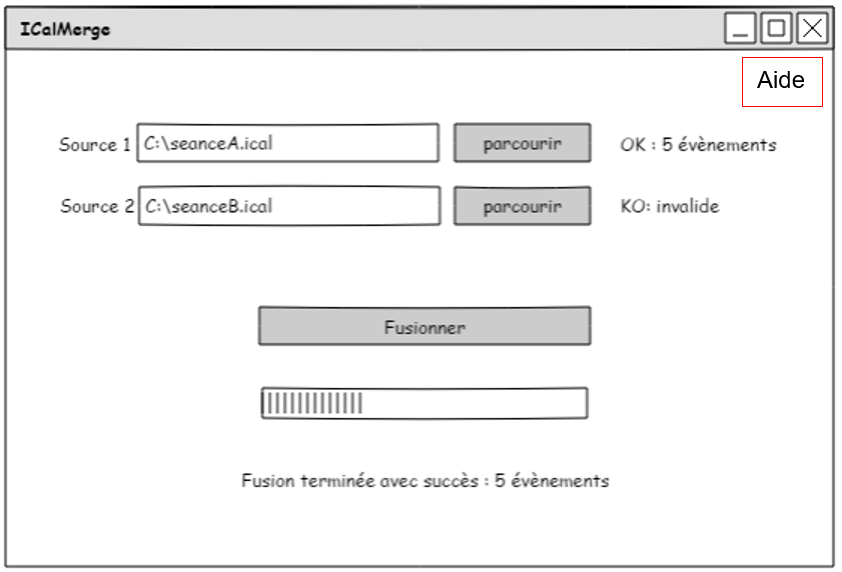
Pour le système de versioning j’ai décidé d’utiliser un git.

J’y accéderai via le logiciel GitHub Desktop.

Le git s’appelle : ICalMerge\_TPI

Voici le lien pour y accéder :<https://github.com/Fwai/ICalMerge_TPI>

Je l’utiliserai tout au long du projet car il me servira à stocker le programme et sa documentation.

1. **Rubrique d’aide**La rubrique d’aide sert à ce que l’utilisateur comprenne assurément comment utiliser l’application.  
     
   Elle sera accessible via un label « Aide » situé en haut à droite de l’interface.  
     
     
   Pour ouvrir le formulaire d’aide, il suffit de cliquer sur « Aide »  
   Cela aura pour conséquence d’afficher un formulaire qui contiendra plusieurs informations sur les différents composants de l’interface.
2. **Explication du format ICal**
3. **Respect des normes ETML**

*Le concept complet avec toutes ses annexes:*

*Par exemple :*

* *Multimédia: carte de site, maquettes papier, story board préliminaire, …*
* *Bases de données: interfaces graphiques, modèle conceptuel.*
* *Programmation: interfaces graphiques, maquettes, analyse fonctionnelle…*
* *…*

## Stratégie de test

*Décrire la stratégie globale de test:*

* *types de des tests et ordre dans lequel ils seront effectués.*
* *les moyens à mettre en œuvre.*
* *couverture des tests (tests exhaustifs ou non, si non, pourquoi ?).*
* *données de test à prévoir (données réelles ?).*
* *les testeurs extérieurs éventuels.*

## Risques techniques

* La barre de progression  
  Je n’ai encore jamais fait de barre de progression auparavant sur windows forms. Il se pourrait que je peine à trouver la manière dont je dois l’utiliser. Il faudrait que je me documente davantage pour être à l’aise avec.

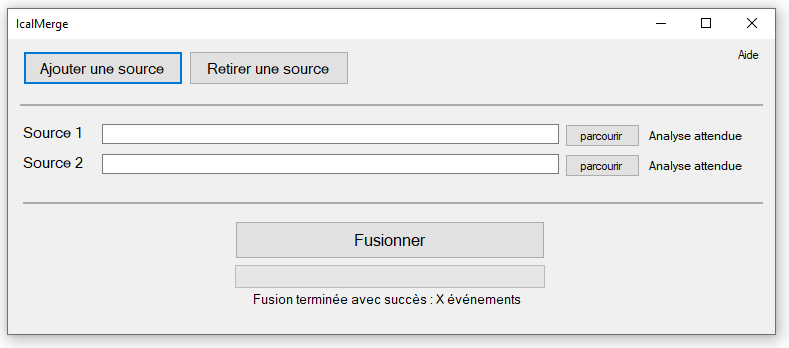
## Planification

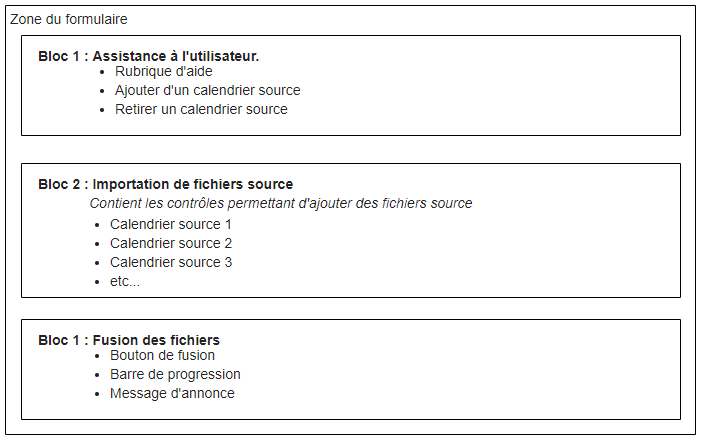
*Révision de la planification initiale du projet :*

* *planning indiquant les dates de début et de fin du projet ainsi que le découpage connu des diverses phases.*
* *partage des tâches en cas de travail à plusieurs.*

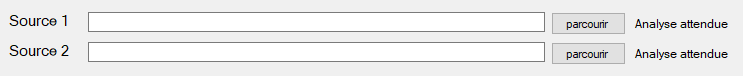
*Il s’agit en principe de la planification* ***définitive du projet****. Elle peut être ensuite affinée (découpage des tâches). Si les délais doivent être ensuite modifiés, le responsable de projet doit être avisé, et les raisons doivent être expliquées dans l’historique.*

## Dossier de conception L’interface se base entièrement sur le mocku-up fourni dans le cahier des charges. Néanmoins, pour des raisons de confort d’utilisation et de fonctionnalités supplémentaires requises, certains contrôles ont été ajoutés.

Voici l’interface graphique crée :

L’interface est composée de trois blocs. Chacun contient des contrôles qui servent à un but précis. Ils se composent ainsi :

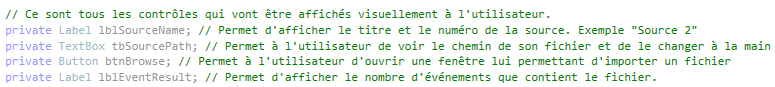
Ces blocs sont définis par des panels. Cela permet de changer la disposition des contrôles par groupes en ayant le minimum d’impact possible sur les autres blocs.

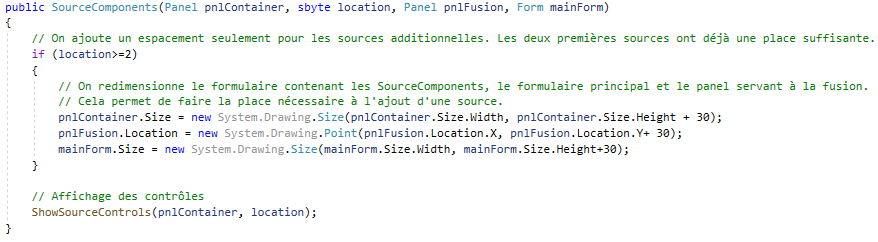
1. **Maximum de 10 fichiers source**Lorsque l’application démarre, il créer deux emplacements de fichiers source. Voici à quoi ils ressemblent :

Pour arriver à ce résultat, nous créons deux objets nommés « SourceComponents ». Chaque objet de cette classe regroupe les contrôles nécessaires à l’affichage**.**

Cette classe permet à l’utilisateur d’ajouter un calendrier à fusionner.

Un objet permet d’ajouter une source. Donc plusieurs objets, implique plusieurs sources.

**Composition d’un SourceComponents :**  
Variables principales :

Le constructeur de la classe :

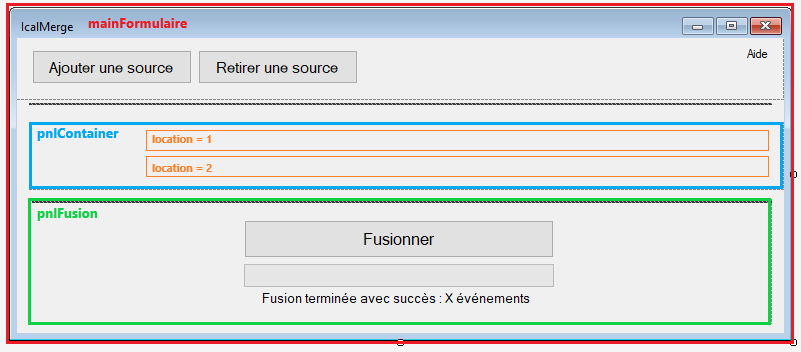
**pnlContainer** : Définit le panel qui va contenir les sources.

**location** : Définit le numéro de la source. Si le programme contient trois sources, en ajouter une nouvelle lui donnera location = 4.

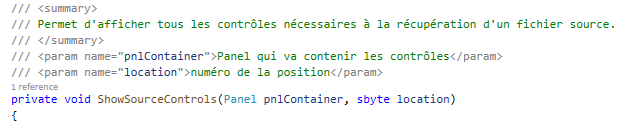
**pnlFusion** : Définit le panel qui contient les contrôles permettant à l’utilisateur de fusionner les fichiers. On en a besoin car, à fois que l’on veut ajouter un fichier source, il faudra baisser ce panel.

**mainForm**: Définit le formulaire principal. Car à chaque fois que l’on veut ajouter une source, il faudra agrandir le formulaire principal.

Voici une représentation graphique d’à quoi correspondent ces valeurs :



La méthode ShowSourceControls :



Cette méthode est appelée par le constructeur. Son contenu pourrait être directement dans le constructeur, mais cela permet de séparer les différentes étapes du processus.

Elle sert à afficher tous les contrôles nécessaires pour que l’utilisateur puisse rajouter une source. Ces contrôles sont stockés dans la classe « SourceComponents ». c’est ceux qui ont été montrés à la page précédentes sous « Variables Principales ».

Voici les contrôles :



**Limitation à 10 fichiers sources :**

Un fichiers source correspond à un SourceComponents.

Pour cela, le formulaire principal contient une liste de Sourcecomponents.

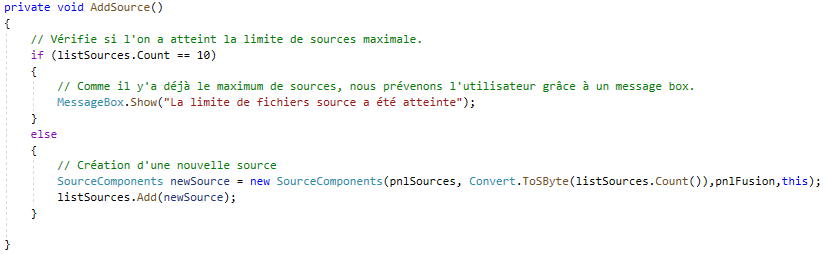
Lorsque l’on clique que le bouton « ajouter une source » :

Si le nombre de sources disponibles actuelles est inférieur à 10

* Créer un Sourcecomponents et l’ajoute à la liste

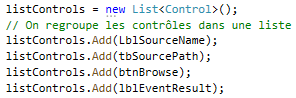
Si le nombre de sources est de 10

* Affiche un message d’erreur avertissant l’utilisateur que le nombre maximal a déjà été atteint

C’est la méthode « AddSource » qui s’occupe de cela.

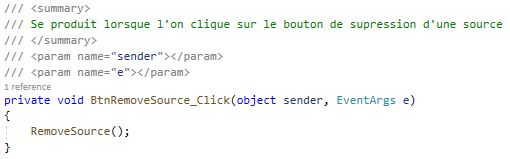
Comme il est possible d’ajouter des sources, il devrait aussi être possible d’en supprimer.

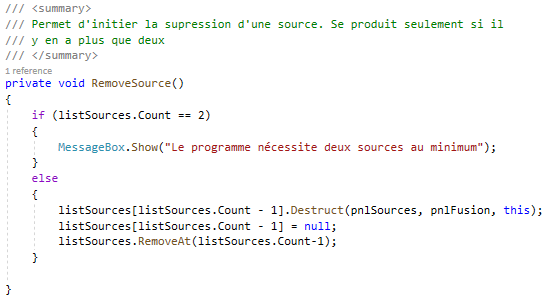
Pour cela j’ai ajouté une liste qui contient les contrôles de la classe SourceComponents. Cette liste se remplit à la fin de la méthode « ShowSourceControls ».



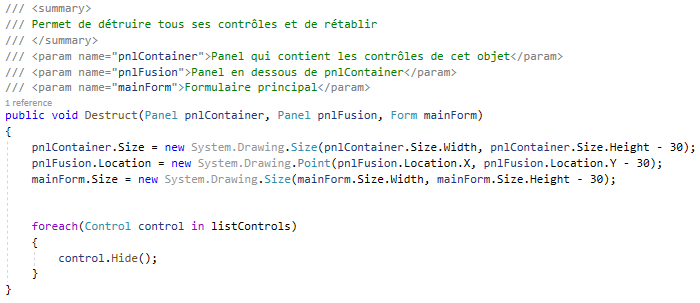
Cela permettra de faire plusieurs actions sur chaque contrôle en une fois. Cela à l’aide d’une boucle « foreach ».

Le bouton « Retirer une source » permet d’effacer une source.

L’événement clic va appeler une méthode nommée « RemoveSource ».

La méthode RemoveSource permet d’initier la suppression d’une source :

S’il y’a plus que deux sources, on lance la méthode « Destroy » du dernier Sourcecomponents contenu dans la liste de sources. Après cela, nous retirons l’objet concerné à la liste. Pour définir le dernier élément, nous prenons le nombre d’objet dans la liste moins un.

La méthode « Destruct », redimensionne les contrôles du formulaire principal. Le but est d’effacer la place qui était nécessaire à l’emplacement de la source. Ensuite une boucle « foreach » permet de cacher tous les éléments visuels. Car l’utilisateur considère ne plus en avoir besoin.

1. **Chargement de fichiers**
2. **Vérification du format automatique**
3. **Résumé avec le nombre d’événements**
4. **Pop-up de fusion**
5. **Barre de progression**
6. **Avertissement fichier de destination**
7. **Vérification de l’intégrité du fichier fusionné**
8. **Utilisation d’un système de versioning**
9. **Rubrique d’aide**
10. **Respect des normes ETML**

*Fournir tous les document de conception:*

* *le choix du matériel HW*
* *le choix des systèmes d'exploitation pour la réalisation et l'utilisation*
* *le choix des outils logiciels pour la réalisation et l'utilisation*
* *site web: réaliser les maquettes avec un logiciel, décrire toutes les animations sur papier, définir les mots-clés, choisir une formule d'hébergement, définir la méthode de mise à jour, …*
* *bases de données: décrire le modèle relationnel, le contenu détaillé des tables (caractéristiques de chaque champs) et les requêtes.*
* *programmation et scripts: organigramme, architecture du programme, découpage modulaire, entrées-sorties des modules, pseudo-code / structogramme…*

***Le dossier de conception devrait permettre de sous-traiter la réalisation du projet !***

# Réalisation

## Dossier de réalisation

La méthode gestion de projet est : Kanban

* *les répertoires où le logiciel est installé*
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*
* *le numéro de version de votre produit !*
* *programmation et scripts: librairies externes, dictionnaire des données, reconstruction du logiciel - cible à partir des sources.*

*NOTE : Evitez d’inclure les listings des sources, à moins que vous ne désiriez en expliquer une partie vous paraissant importante. Dans ce cas n’incluez que cette partie…*

## Description des tests effectués

*Pour chaque partie testée de votre projet, il faut décrire:*

* *les conditions exactes de chaque test*
* *les preuves de test (papier ou fichier)*
* *tests sans preuve: fournir au moins une description*

## Erreurs restantes

*S'il reste encore des erreurs:*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

## Liste des documents fournis

*Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions*

* *le rapport de projet*
* *le manuel d'Installation (en annexe)*
* *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
* *autres…*

# Conclusions

*Développez en tous cas les points suivants:*

* *Objectifs atteints / non-atteints*
* *Points positifs / négatifs*
* *Difficultés particulières*
* *Suites possibles pour le projet (évolutions & améliorations)*

# Annexes

## Résumé du rapport du TPI / version succincte de la documentation

## Sources – Bibliographie

C# Tutorial – Drag and Drop Text files into a RichTextBox ¦ FoxLearn

Lien : <https://www.youtube.com/watch?v=RCnHLu0sp5c>

Extracting Path from OpenFileDialog path/filename  
Lien : <https://stackoverflow.com/questions/439007/extracting-path-from-openfiledialog-path-filename>

*Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)… Et de toutes les aides externes (noms)*

## Journal de travail

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Durée** | **Activité** | **Remarques** |
|  |  |  |  |
|  |  |  |  |

## 

## Manuel d'Installation

## Manuel d'Utilisation

## Archives du projet

*Media, … dans une fourre en plastique*