

# PROJET GÉNIE LOGICIEL

## Rapport par Simon Audrix et Gautier Laisné

---

<b>Introduction</b>	<b>1</b>
<b>Spécifications</b>	<b>2</b>
Analyse fonctionnelle	2
Analyse des besoins pour l'interface avec l'utilisateur	2
Schémas UML	2
Architecture de l'application	2
Justification des choix	3
Répartition des tâches dans le groupe	3
<b>Résultats et tests</b>	<b>4</b>
Listes des exigences métier respectées par le programme	4
Copies d'écran illustrant le fonctionnement	4
Description des protocoles de tests	4
Résultats des tests	5
<b>Bilan et perspectives</b>	<b>6</b>
Points positifs	6
Points négatifs	6
Evolutions possibles	6

---

---

## Introduction

Dans ce projet, nous devons concevoir une application de gestion de CV. Celle-ci permettra à un utilisateur de créer et modifier des CV qu'il a saisi. L'intérêt de cette application réside dans le fait de centraliser des éléments de *carrière* (autrement dit, un événement professionnel intéressant) et de pouvoir les référencer afin de les utiliser rapidement lors de la création d'un nouveau CV.

La création de CV est une étape pénible et coûteuse en temps. Faciliter sa création est un besoin non négligeable.

## Spécifications

### Analyse fonctionnelle

Une des premières parties de notre projet a été de se concentrer sur les fonctionnalités attendues pour répondre aux exigences métiers du projet. Celles-ci peuvent se déduire à partir des User Journeys fournis. Voici les fonctionnalités que nous avons déduites :

- Ajout/modification/suppression d'un CV par un utilisateur
- Ajout/modification/suppression d'une rubrique par un utilisateur
- Ajout/modification/suppression d'un élément par un utilisateur
- Organisation des rubriques dans le CV
- Sauvegarde d'un CV et de son contenu
- Accès aux précédents CV sauvegardés
- Convertisseur d'un CV vers un format spécifique entre HTML et PDF

### Analyse des besoins pour l'interface avec l'utilisateur

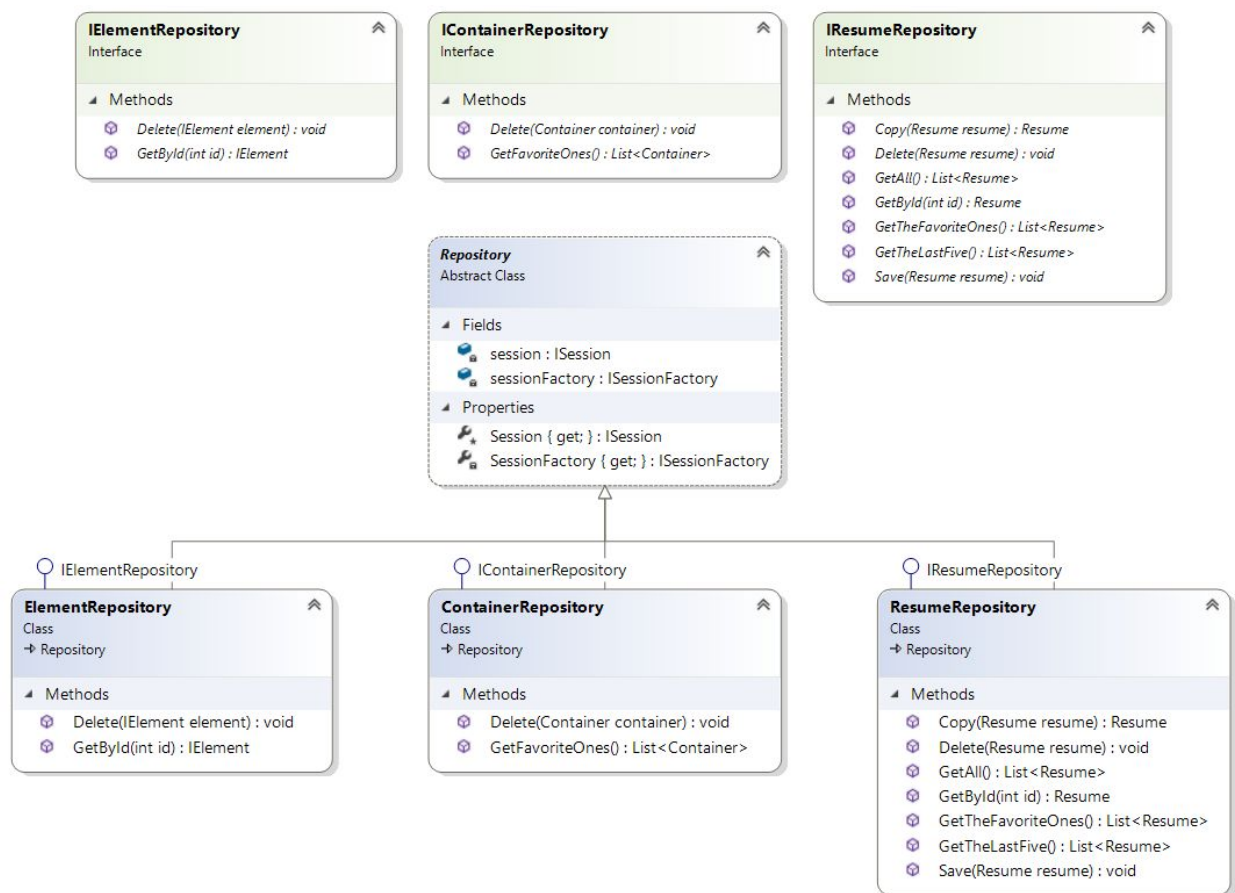
Nous avons évalué les interactions possibles avec l'utilisateur pour en arriver aux conclusions suivantes :

- il y a 2 principales fenêtres : l'écran d'accueil, permettant à l'utilisateur de choisir un CV, d'en créer un, d'en dupliquer un, d'en mettre un en favoris et/ou d'en supprimer

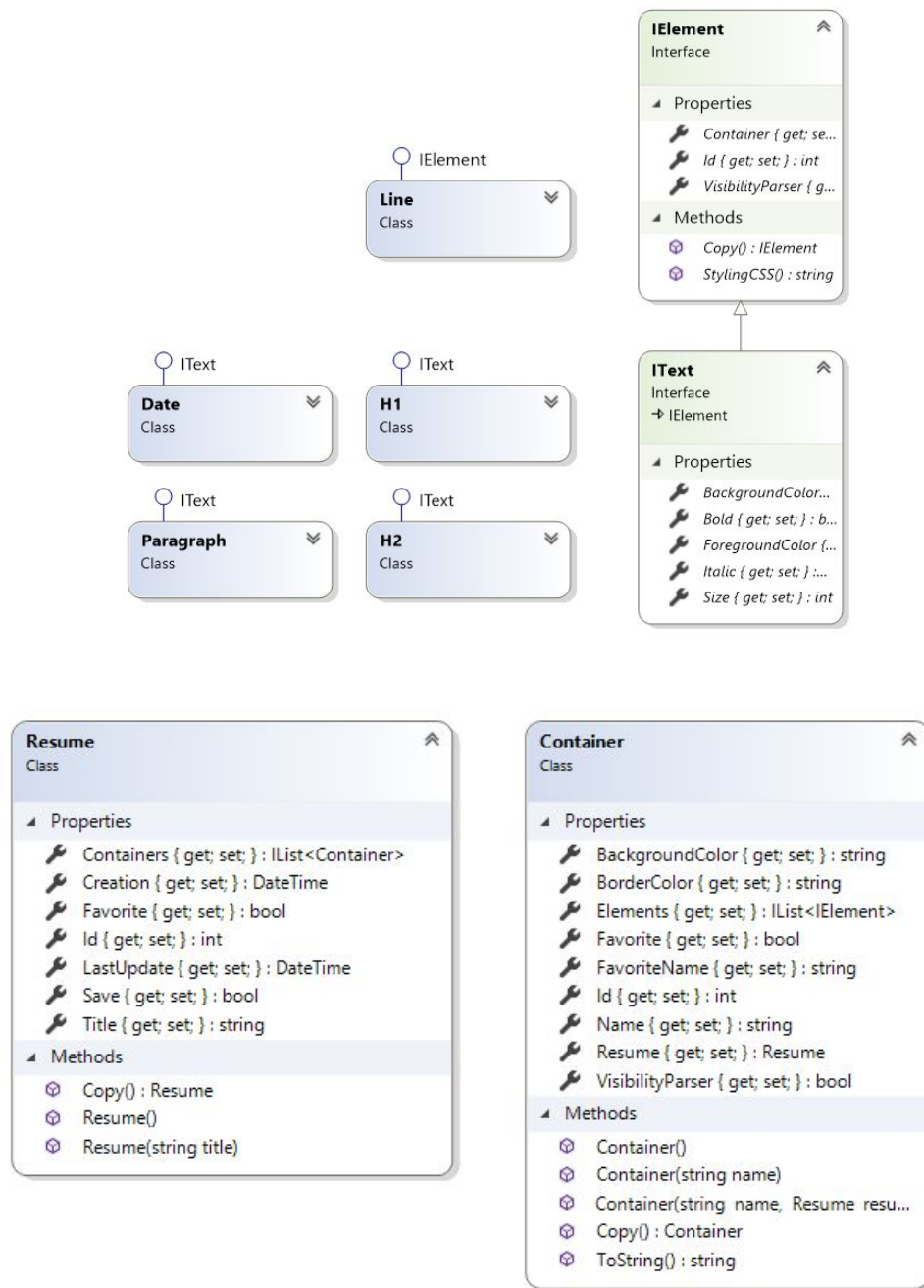
un. Et l'écran d'édition, où l'utilisateur peut, via des glissés-déposés, ajouter des rubriques au CV et des éléments aux rubriques.

- Le CV à modifier est au centre de l'interface : toutes les modifications se font autour.
- Le contenu d'un CV se basera sur les éléments familiers qui structurent les documents. On a donc décidé de se concentrer sur des cas spécifiques en associant une classe métier à chacune :
  - les titres (H1)
  - les sous-titres (H2)
  - les zones de texte (Paragraph)
  - les éléments de date (Date)

## Schémas UML



*Diagramme des classes du sous-projet DAL*



### Diagramme des classes du sous-projet Domain :

Un Resume contient une liste de Containers et chaque Container sait qui est son parent.

Un Container contient une liste de IElement. Chaque IElement connaît son parent.

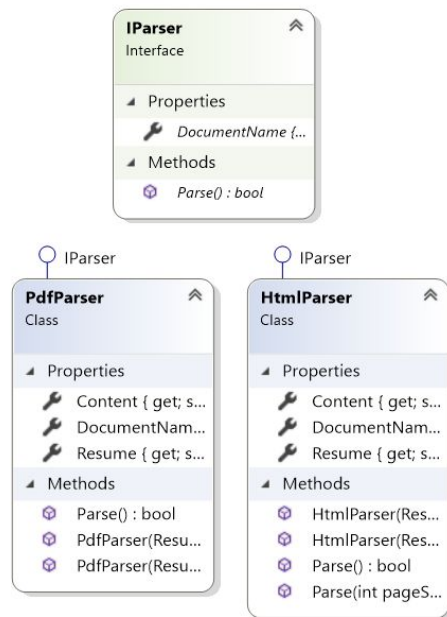


Diagramme des classes du sous-projet Domain pour les convertisseurs de CV

## Architecture de l'application

Notre application est architecturée en 3-couche avec les namespaces suivants :

- **Domain** regroupe les classes métiers
- **DAL** gère les échanges entre la base de données et les classes métiers
- **App** s'occupe de tout ce qui est interactions et le visuel de l'application. On a de plus une architecture MVP dans ce sous-projet.

## Justification des choix

Nous avons pris une structure en couche pour réaliser l'accès aux données car après l'avoir vu en TD elle nous semblait la plus simple et la plus logique à mettre en place.

La construction de l'application a en grande partie été réalisée en MVP (Model View Presenter). Bien que ce design pattern semble un peu beaucoup pour cette simple application, il nous semblait intéressant de le tester dans un cadre scolaire afin de découvrir son fonctionnement et ses possibilités.

---

## Répartition des tâches dans le groupe

Globalement, la répartition a été nette. Il y a eu une première semaine en commun dédiée à la conception et aux choix de réalisation, des fonctionnalités demandées et de prise de connaissance de l'ampleur du projet. Puis nous nous sommes séparés comme suivant :

- Gautier s'est occupé de l'implémentation des classes métiers, de la persistance des données avec NHibernate (mapping, interfaces et repository) ainsi que des convertisseurs de CV vers HTML et PDF.
- Simon a géré tout l'aspect graphique avec Winforms sous MVP. Celle-ci a été pensée de telle sorte qu'elle soit utilisable avec des glissés-déposés. Il a utilisé un stub afin de générer des données fictives et pouvoir travailler avec les implémentations du DAL.

---

## Résultats et tests

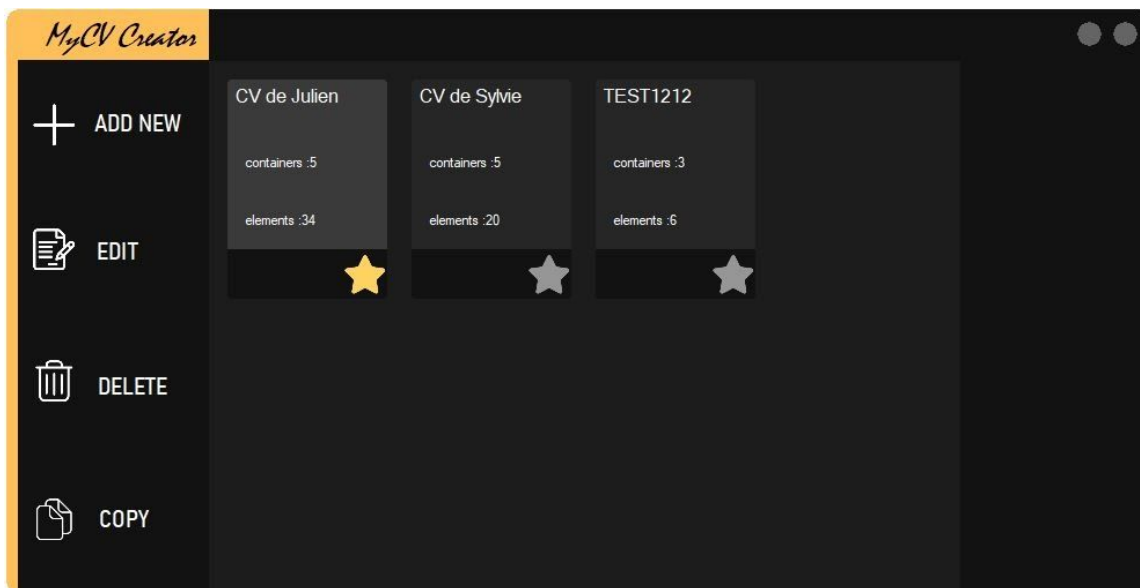
### Listes des exigences métier respectées par le programme

Les exigences suivantes sont respectées par le programme :

- Créer un nouveau CV
- Ajout d'une nouvelle rubrique au CV actuellement ouvert
- Ajout d'un élément de carrière au CV actuellement ouvert
- Suppression d'un élément de carrière
- Association d'un élément de carrière à une rubrique
- Edition d'un élément de carrière
- Suppression d'une rubrique
- Possibilité de renommer une rubrique
- Sélection des éléments et rubriques qui figureront dans le CV
- Générer un CV au format PDF
- Générer un CV au format web HTML/CSS

### Copies d'écran illustrant le fonctionnement

Lorsque l'application se lance, l'utilisateur se retrouve directement sur le menu:

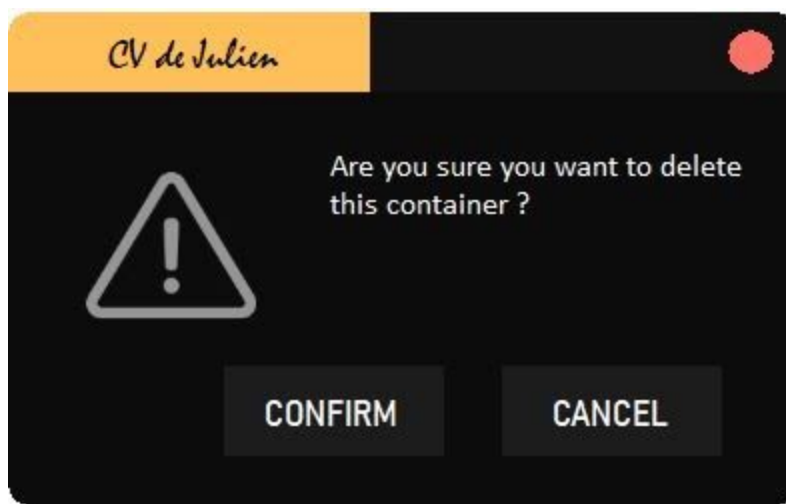


---

Sur ce menu, il peut visualiser au centre ses CV précédemment enregistrés. Les ajouter en tant que favori ou les sélectionner.

Sur la gauche il a accès à différents boutons d'action qui lui permettront de supprimer ou de copier un CV ou de se rendre dans l'éditeur en ajoutant un CV vierge ou en sélectionnant un CV à éditer.

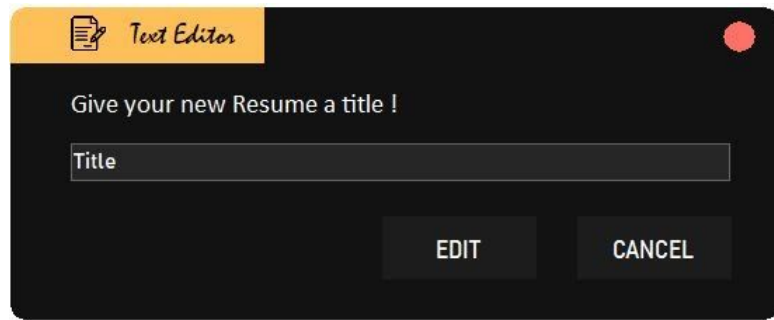
Sur cet écran, les comportements sont contrôlés, en effet une confirmation sera demandée à l'utilisateur pour toute demande de suppression ou de copie d'un CV.



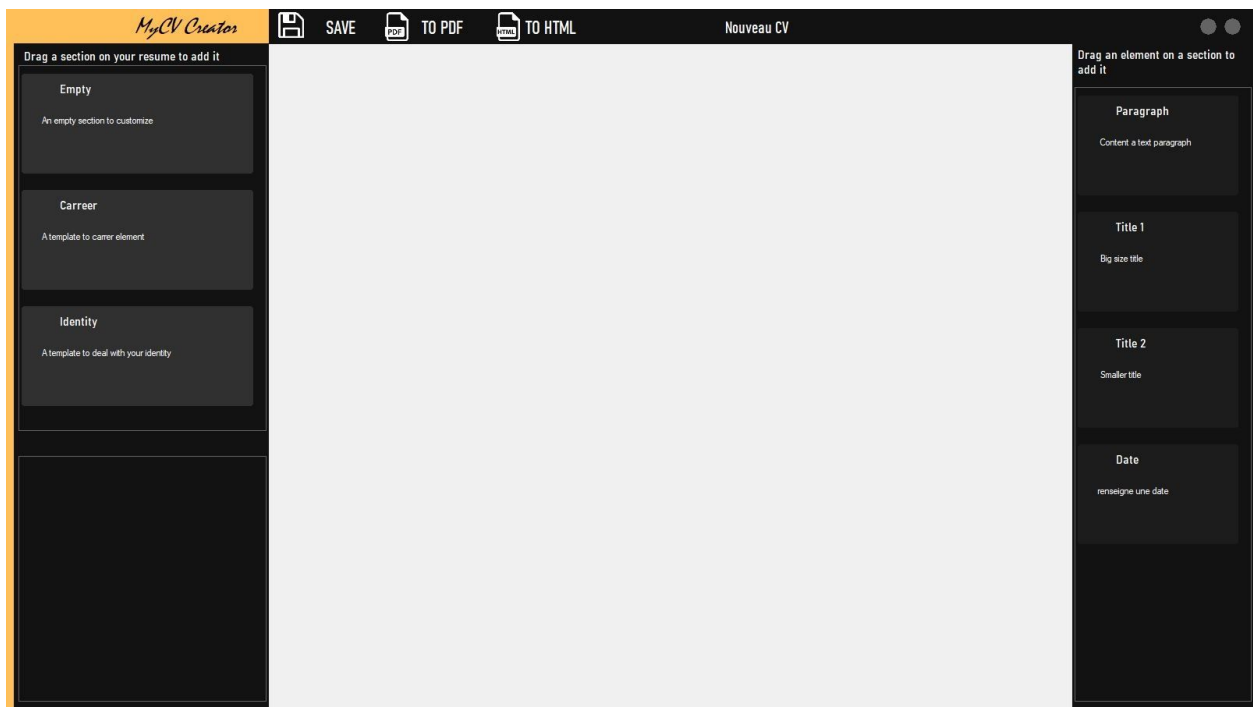
Cela permet d'éviter à l'utilisateur de commettre d'irréparables erreurs !



Si l'utilisateur choisit d'ajouter un nouveau CV on lui propose alors de renseigner le nom de son nouveau CV grâce à un éditeur de texte intégré dans l'application.

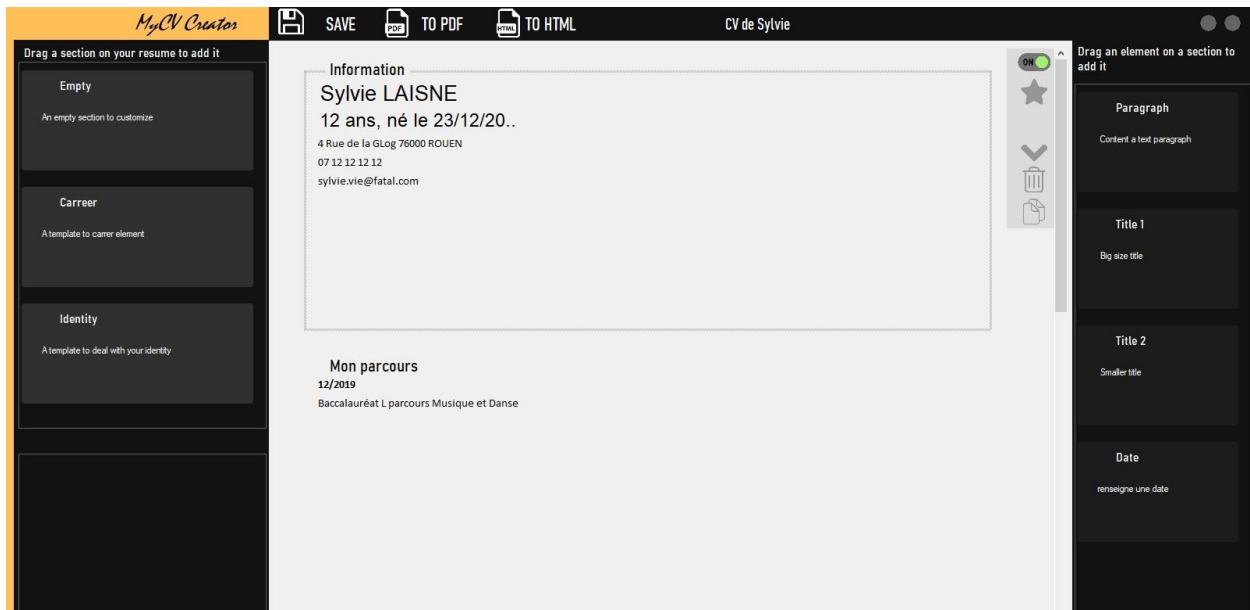
A dark-themed dialog box titled "Text Editor" with a red close button in the top right corner. The main text inside says "Give your new Resume a title !". Below this is a text input field with the placeholder text "Title". At the bottom right, there are two buttons: "EDIT" and "CANCEL".

et il se retrouve alors devant l'écran d'édition:



Ce dernier dispose d'une zone vierge au centre représentant le nouveau CV. Dans la colonne de gauche l'utilisateur dispose de ses régions et à droite de ses éléments. Il peut alors les déposer dans l'éditeur pour commencer à remplir son CV.

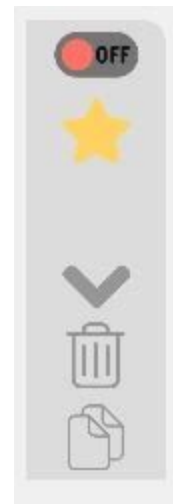
Dans le cas où il ajoute des éléments, ou choisit d'éditer un ancien CV, il se retrouve alors avec son éditeur rempli :



Il dispose alors d'un grand nombre d'interaction. Il peut sélectionner une section ou un élément afin d'afficher son panel de contrôles.

Il peut alors éditer les propriétés de chaque élément de son CV. Pour une section il peut respectivement sélectionner :

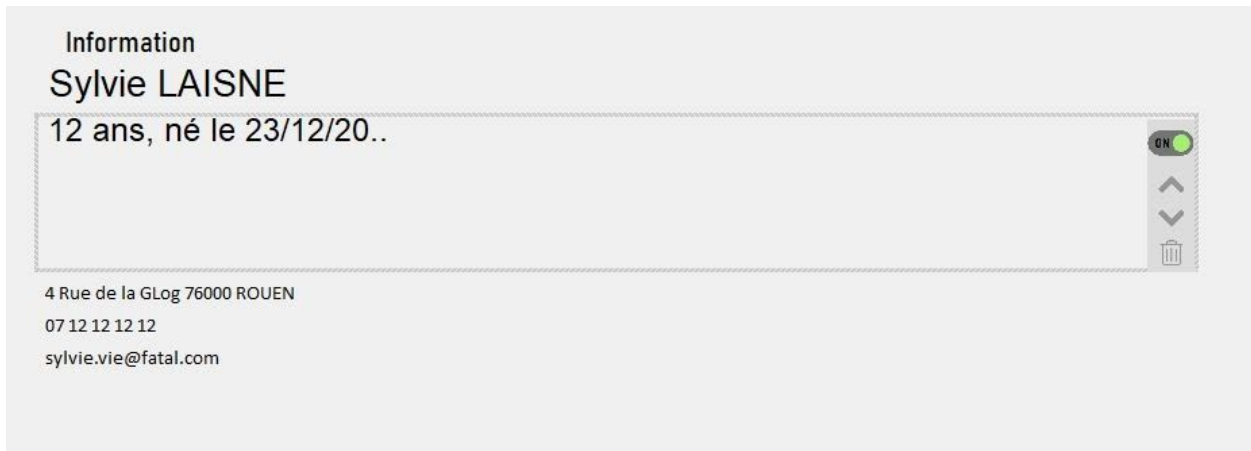
- Son apparition dans les parses
- Son état favori
- Sa position (Haut - Bas)\*
- Le supprimer
- Le copier



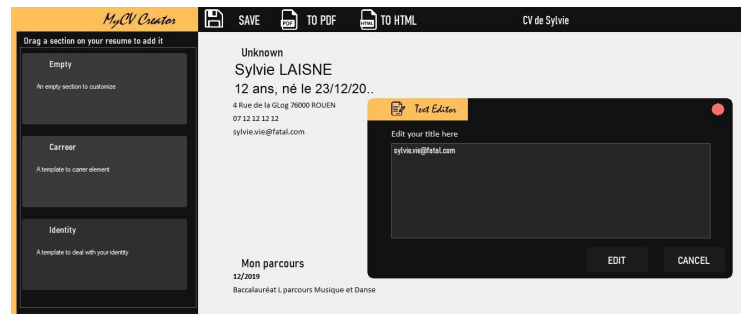
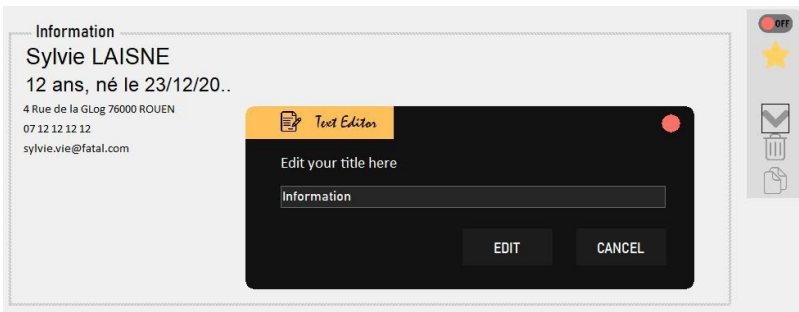
Encore un fois, les actions de suppression et d'édition sont soumises à confirmation.

Pour les éléments, les actions sont limitées, l'utilisateur ne pourra en effet pas le mettre en favori ni le copier.

\* Selon la position de l'élément il se peut que certaines interactions ne soit pas disponibles. En effet l'élément du haut ne peut pas être déplacé plus haut encore !



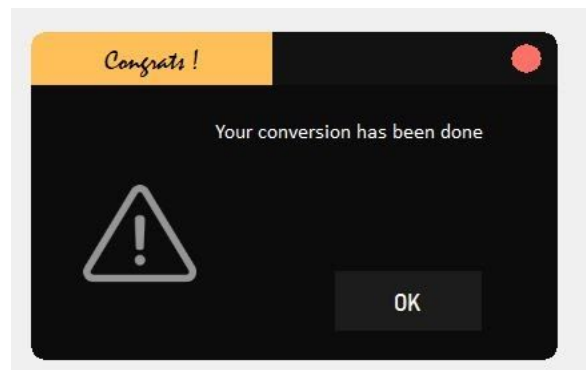
Chaque élément peut donc être sélectionné. On peut ensuite double cliquer sur son contenu pour l'éditer. En fonction de l'élément on se retrouve donc face à trois éditeur différents qui fonctionnent tous de la même



manière et permettent d'éditer différemment les contenus des éléments. Ici, on aperçoit le éditeur de textes court et long. Il existe aussi un éditeur de date spécifique pour correctement formater les entrées de

l'utilisateur. Ces affichages permettent aussi d'éditer le titre des section et le titre du CV toujours en double cliquant dessus.

D'autre fonctionnalité importantes sont disponibles dans l'éditeur celle de sauvegarder évidemment mais aussi la possibilité d'exporter le CV en pdf et en html grâce au bouton disposé



---

dans la barre supérieur. Lors de l'utilisation de ces derniers les utilisateurs recevront un feed back leur permettant de connaître le résultat de leur action.

## Description des protocoles de tests

Pour la persistance des données, des tests ont été fait en créant des tests unitaires. Cependant ceux-ci n'ont pas respecté à la lettre le principe des tests unitaires car une fois une méthode validée (par exemple la recherche d'un élément dans un CV particulier), des données utilisées dans cette méthode peuvent être supprimées et ainsi la rendre non valide lors de re-testing (par exemple en supprimant le CV). Par contre cela a permis de bien cadrer le code et de ne pas se servir de la console (la plupart des retours visuels se faisant dans *phpMyAdmin* en allant directement fouiller dans la BDD).

Ce qui a été vérifié à travers ces méthodes sont :

- l'insertion de données
- la mise à jour de données quelconque
- les cascades :
  - supprimer un CV supprime tous ses enfants
  - sauvegarder un tout nouveau CV insère ses enfants dans la BDD
  - modifier un CV et les impacts sur les enfants
- la récupération de données (avec et sans tri)

De la même manière, pour les convertisseurs de CV voici ce qui a été testé:

- la conversion en HTML de 2 CV différents
- l'affichage (ou le non affichage) des éléments visibles dans les fichiers convertis
- les mêmes tests précédents ont été fait pour le PDF

Tous ces tests se trouvent dans le sous-projet **DALTests**.

## Résultats des tests

Les tests, en suivant un scénario assez précis, ont tous été validés par l'explorateur de tests, en base de données et visuellement sur notre interface.

---

## Bilan et perspectives

### Points positifs

Voici ce que nous avons retiré de positif concernant le projet :

- Habitude à NHibernate et à Winforms
- Notre vision de la programmation pour interfaces a mûri et nous avons pris du recul là dessus
- Les exigences ont été traitées dans la majorité des cas
- Nous avons pu appréhender les fonctionnalités de Drag&Drop en winforms

### Points négatifs

Comme tout projet, il y a des hauts et des bas.

- Nous avons passé trop de temps sur des choses qui ne sont pas utiles en début de projet (le *drag and drop* fût compliqué à mettre en place tout comme l'architecture MVP qui a complexifié le code)
- Nous avons prévus un grand nombre de fonctionnalités qui n'ont pas pu toutes être implémentées étant obligés de commencer par les exigences directement liées au projet. Un temps plus long nous aurait permis de créer un éditeur bien plus abouti ce qui crée une légère frustration. Nous restons cependant très contents du travail fourni.

### Evolutions possibles

Nous avons pensé dès le début à incorporer la personnalisation de styles afin que l'utilisateur puisse choisir s'il veut son texte en italique, en gras, en souligné, plus grand, plus petit etc... Ceci explique certaines propriétés de nos éléments. De part un manque de temps, nous n'avons pas pu implémenter cette fonctionnalité mais c'est une

---

Pour les dates, il y a des améliorations à faire, car nous n'avions pas pris en compte la notion d'intervalle de temps. Ainsi, nous nous retrouvons coincé devant un CV ne proposant que le début des événements de carrière.

Graphiquement, une lourde tâche a été entamée, il faudrait désormais peaufiner le design et tester sur des utilisateurs afin de la faire progresser de la meilleure manière possible.

Il reste également un grand nombre d'implémentation qui pourrait venir s'ajouter pour rendre plus agréable l'expérience utilisateur et améliorer l'ergonomie du code comme:

- Du feedback lors des opérations de glisser déposer
- Une demande de confirmation d'enregistrement lors de la fermeture de la fenêtre d'édition
- Une possibilité de retour à l'accueil (via un bouton ou lors de la fermeture de la fenêtre d'édition)
- L'implémentation du glisser déposer via des interface pour les objet de types *"Droppables"* ( Permettrait une plus grande flexibilité et propreté du code)
- Une généralisation de la méthode de rendu du CV
- Un rendu seulement pour les élément modifiés grâce à une gestion d'événement personnalisé de modification
- L'accès à un plus grand nombre d'élément
- L'ajout de la modification de style
- Les éléments de style permettant d'améliorer le design de son CV
- L'ajout de bouton d'export en PDF et en HTML pour chaque CV dès l'accueil à côté du bouton de favori
- Une simplification de l'encapsulation des objets métier pour faciliter leur modification via l'interface graphique (modification unique pour l'élément graphique et l'objet backend pour éviter la duplication de code et la lourdeur d'exécution)

Plein d'autre modification pourrait être envisagé comme notamment la gestion précise du placement des objets par l'utilisateur pour lui permettre un plus grand contrôle sur la personnalisation de son CV. Cependant cela nécessiterait un projet beaucoup plus long...