

RAPPORT DE PROJET

Projet : Gestionnaire de listes de tâches

COUTABLE Guillaume, RULLIER Noémie  
14 février 2013



# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Les fonctionnalités</b>	<b>3</b>
2.1	Fonctionnalités principales . . . . .	3
2.2	Fonctionnalités secondaires . . . . .	3
<b>3</b>	<b>Storyboard, Paper Prototype et Scénarios</b>	<b>4</b>
3.1	Storyboard . . . . .	4
3.2	Paper Prototype . . . . .	5
3.2.1	Les maquettes . . . . .	5
3.2.2	Choix réalisés . . . . .	6
3.3	Scénarios . . . . .	6
3.3.1	Scénario 1 - Crédit/Création/Utilisation/Suppression de liste . . . . .	6
3.3.2	Scénario 2 - Utilisation des templates . . . . .	7
<b>4</b>	<b>L'IHM</b>	<b>8</b>
4.1	Le menu . . . . .	8
4.2	Boîte à outils . . . . .	8
4.3	Zone d'affichage de la liste . . . . .	9
<b>5</b>	<b>Le modèle</b>	<b>10</b>
<b>6</b>	<b>Limite de l'application</b>	<b>11</b>
<b>7</b>	<b>Conclusion générale</b>	<b>12</b>

## 1 Introduction

L'objectif de ce projet fut de développer un gestionnaire avancé de tâches. Celui-ci devait permettre de créer des listes de tâches et de suivre l'avancement de celles-ci.

Afin de créer cette application que nous avons appelé *Taskinator*, nous avons du établir plusieurs étapes d'avancement du projet. Ce rapport présentera ces étapes les unes après les autres (même si lors de ce projet certaines étapes se sont croisées).

## 2 Les fonctionnalités

La première étape fut d'analyser l'ensemble des fonctionnalités que notre application devait proposer.

### 2.1 Fonctionnalités principales

Voici dans un premier temps les fonctionnalités principales :

**Créer une liste :** cette fonctionnalité permet à l'utilisateur de créer une liste vide.

**Créer une liste ordonnée :** cette fonctionnalité permet à l'utilisateur de créer une liste ordonnée vide. L'ensemble des éléments de cette liste doivent être effectué dans un ordre précis.

**Créer une tâche :** cette fonctionnalité permet à l'utilisateur de créer une tâche.

**Supprimer un élément :** cette fonctionnalité permet de supprimer une tâche ou une liste (ordonnée ou non). Cette fonctionnalité est à manipuler avec précaution, en effet dans le cas d'une liste, la suppression de celle-ci implique aussi la suppression de tous ses éléments (listes ou tâches).

**Enregistrer :** cette fonctionnalité permet à l'utilisateur d'enregistrer sa liste dans un document sur son disque dur.

**Enregistrer un template :** cette fonctionnalité permet à l'utilisateur d'enregistrer la liste qu'il vient de créer comme un template afin que la structure de celle-ci soit réutilisable.

**Ouvrir un template :** cette fonctionnalité permet à l'utilisateur de créer une liste à partir d'un template enregistré. Il devra cependant renseigné le nom de cette liste ainsi que toutes les dates de tous les éléments. Il peut ensuite continuer à modifier cette liste.

### 2.2 Fonctionnalités secondaires

Voici les fonctionnalités secondaires :

**Paramètre :** cette fonctionnalité permet à l'utilisateur de modifier le type de l'élément sélectionné. Il pourra par exemple choisir de modifier une liste en liste ordonnée ou en une tâche. Cette fonctionnalité est à manipuler avec précaution, en effet si l'utilisateur décide de transformer une liste en tâche l'ensemble des éléments de la liste seront supprimés.

**Monter / Descendre :** cette fonctionnalité permet de monter ou descendre un élément dans l'arborescence de la liste. Dans le cas d'une liste, tous ces éléments sont aussi monté/descendu d'un rang. Si le changement se fait au sein d'une liste ordonnée l'ordre des éléments est aussi changé. A chaque déplacement, une vérification de la cohérence des dates est effectuée et l'utilisateur en est informé.

**Historique :** cette fonctionnalité permet d'annuler ou rétablir des actions faites par l'utilisateur.

**Gérer ses templates :** cette fonctionnalité permet à l'utilisateur de supprimer les templates qu'il a enregistré.

### 3 Storyboard, Paper Prototype et Scénarios

#### 3.1 Storyboard

Le storyboard permet de montrer à quoi sert l'application. Il utilise les star people de Bill VerPlank. A la fin du storyboard, le personnage atteint son but et est satisfait. Nous avons donc ici créer notre storyboard pour notre application :

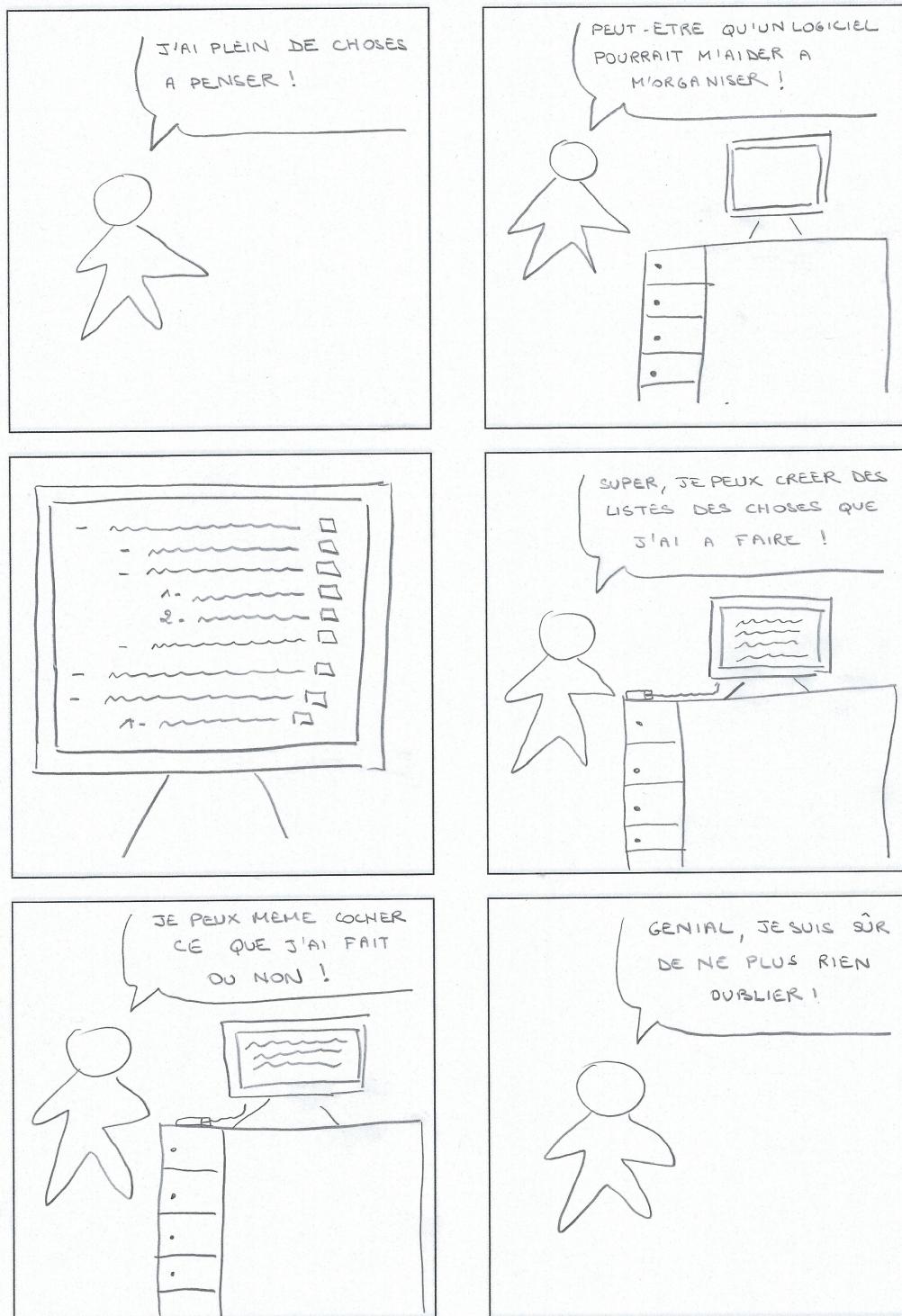


FIGURE 1 – Le storyboard

## 3.2 Paper Prototype

### 3.2.1 Les maquettes

Voici quelques unes des maquettes réalisée pour le paper prototype. Nous expliquons dans la section suivante quels ont été les choix effectués. Les maquettes ici présentent sont celles qui ont été utilisées pour la réalisation de la vidéo du scénario 1 présenté ci-dessous.



FIGURE 2 – Le paper prototype

### 3.2.2 Choix réalisés

Nous avons du faire quelques choix pour la conception de l'interface. Nous vous les présentons ici et surtout nous expliquons pourquoi nous avons choisi telle solution :

- **Fonctionnalité de création (liste ordonnée ou non et tâche)** : nous avons décider d'adopter un système de sélection de l'élément "parent". Expliquons ce système. Lorsque l'utilisateur créer une liste, un élément vide de cette liste apparaît. L'utilisateur doit alors sélectionner cet élément et définir de quel type il sera. Lorsque cet élément à un type, un nouvel élément vide apparaît ; l'utilisateur peut alors définir un autre élément ... Si l'utilisateur ne souhaite pas continuer à remplir cette liste, il ne tient plus compte des éléments vides.
- **Fonctionnalité Monter et Descendre** : nous avons décider de placer ces fonctionnalités dans la boîte à outils plutôt que sur l'élément à déplacer pour un gain de temps au niveau de l'utilisation. En effet, si nous avions fait le choix de mettre les boutons monter et descendre sur l'élément, lorsque l'utilisateur souhaite déplacer l'élément de plusieurs place, il va perdre le focus de la souris sur ces boutons. Attention, lors de l'utilisation de cette fonctionnalité dans une liste ordonnée ; si on monte ou descend un élément de tel façon que un élément coché se trouve être en dessous d'un élément non coché dans la liste, alors le déplacement ne sera pas activé. De plus, nous avons décidé d'afficher une fenêtre d'avertissement permettant à l'utilisateur d'être informé si il y a une incohérence au niveau des dates des éléments déplacés. Nous avons décidé de mettre en valeur la plus récente des dates (des éléments modifiés) par le remplissage du fond du champs date d'une couleur orange (permet d'attirer l'attention et de prévenir l'utilisateur).
- **Fonctionnalité Supprimer** : nous avons choisi d'ajouter cette possibilité de suppression sur chaque élément pour que l'utilisateur puisse supprimer plus rapidement. Nous avons de plus décider de faire apparaître une fenêtre d'avertissement car la suppression d'une liste sélectionnée peut entraîner la suppression d'éléments. Pour plus de sécurité, le focus de la validation de la suppression est par défaut sur *Non*.
- **Fonctionnalité Paramètre** : nous avons décider pour cette fonctionnalité (qui peut entraîner la suppression de différents éléments) de faire apparaître ici aussi une fenêtre d'avertissement. Dans cette fenêtre le focus de la validation du changement est mise à *Non*. On perd cependant un peu de vitesse d'exécution (si l'utilisateur souhaite effectivement bien effectuer ce changement) au profit de plus de sécurité.
- **Widget désactivé** : nous avons décidé de griser tout menu, bouton inutilisable à un état de l'application afin que l'utilisateur n'exécute pas d'actions impossibles. Cela évitera qu'il se pose des questions s'il voit aucun changement effectué.

## 3.3 Scénarios

Afin de tester notre PaperPrototype, nous avons créer plusieurs scénarios.

### 3.3.1 Scénario 1 - Crédation/Utilisation/Suppression de liste

Ce premier scénario a été utilisé pour tester le paper prototype. Il permet de créer une liste non ordonnée et d'y ajouter une liste ordonnée avec ses propres tâches et des tâches.

1. L'utilisateur choisit quel type de liste il souhaite créer, il choisit ici une liste non ordonnée. Il lui donne un nom et une date de fin. (Pour notre scénario cette liste sera appelée la liste mère)
2. Il crée ensuite une liste ordonnée (qui est le premier élément de la liste mère). Il lui donne un nom et une date. (Pour notre scénario cette liste sera appelée la liste 1)
3. Il créer ensuite une tâche qui sera le premier élément de la liste 1. Il lui donne un nom et une date. (Pour notre scénario cette tâche sera appelée la tâche 1.1)
4. Il créer ensuite une tâche qui sera le deuxième élément de la liste 1. Il lui donne un nom et une date. (Pour notre scénario cette tâche sera appelée la tâche 1.2)
5. Il souhaite maintenant échanger les tâches 1.1 et 1.2, il sélectionne donc la tâche 1.1 et la place au dessous de la tâche 1.2

6. Une popup peut apparaître et informe l'utilisateur que l'action qu'il vient d'effectuer provoque un conflit de date. Le champs date concerné par le conflit prend une couleur orange d'avertissement.
7. L'utilisateur change le type de la liste 1 en une liste non ordonnée.
8. L'utilisateur supprime la tâche 1.2
9. L'utilisateur supprime la tâche 1. Une popup apparaît pour l'avertir que cette suppression supprimera aussi tous les éléments de la liste.

### 3.3.2 Scénario 2 - Utilisation des templates

Ce scénario permet de créer une liste à partir d'un template. De modifier cette liste et de la réenregistrer comme un nouveau template.

1. L'utilisateur choisit de créer une liste à partir d'un template. Il choisit ici le template qui correspond à la préparation d'un cours, puis donne un nom et une date à sa liste.
2. Il va ensuite pour tous les éléments de ce template donner une date.
3. L'utilisateur va ensuite modifier cette liste en y ajoutant une tâche à la liste mère (il lui donne un nom et une date).
4. Il va ensuite vouloir enregistrer cette nouvelle liste comme un nouveau template.

## 4 L'IHM

Afin de créer une interface la plus simple et explicite possible pour l'utilisateur nous avons du effectuer différents choix que nous allons expliquer ci-dessous.

La fenêtre principale est tout d'abord composée d'un menu, d'une boîte à outils et d'une zone réservée à l'affichage de la liste en cours de création.

### 4.1 Le menu

Le menu permet de regrouper différentes fonctionnalités sous un thème/rubrique. On retrouvera les fonctionnalités suivantes :

- Fichier
  - **Nouveau** : qui permet de créer une nouvelle liste vide ou à partir d'un template.
  - **Ouvrir** : qui permet d'ouvrir une liste au format *.tor* enregistrer sur un disque dur de l'utilisateur.
  - **Ouvrir récent** : qui permet à l'utilisateur d'ouvrir l'une des cinq dernières listes ouvertes dans cette application.
  - **Enregistrer** : qui permet à l'utilisateur d'enregistrer sa liste en cours à chemin déjà spécifié.
  - **Enregistrer sous** : qui permet à l'utilisateur de choisir le chemin dans lequel il souhaite enregistrer sa liste et de l'enregistrer.
  - **Exporter** : qui permet à l'utilisateur d'exporter sa liste au format PDF.
  - **Quitter** : qui permet à l'utilisateur de quitter l'application.



FIGURE 3 – Le menu Fichier

- Édition
  - **Annuler** : qui permet à l'utilisateur d'annuler une action qu'il vient d'effectuer.
  - **Rétablir** : qui permet à l'utilisateur de rétablir une action qu'il vient d'annuler.

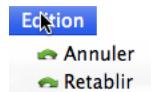


FIGURE 4 – Le menu Fichier

- Outils
  - **Enregistrer template** : qui permet à l'utilisateur d'enregistrer sa liste en tant que template au format *.ulk*.

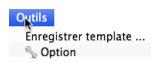


FIGURE 5 – Le menu Fichier

- Aide
  - **A propos** :

### 4.2 Boîte à outils

Cette boîte à outils permet de rassembler toutes les fonctionnalités que l'utilisateur sera amené à utiliser fréquemment. Nous avons décidé de mettre cette boîte à outils à gauche, puisque dans

notre culture le sens de lecture est de gauche à droite. Toutes les fonctionnalités présentent dans cette boîte à outils seront appliquée sur l'élément sélectionné dans la zone d'affichage de la liste. Cette boîte à outils contient les fonctionnalités suivantes :

- **Créer une liste non ordonnée (☰)** : elle permet à l'utilisateur de créer une nouvelle liste non ordonnée.
- **Créer une liste ordonnée (☷)** : elle permet à l'utilisateur de créer une nouvelle liste ordonnée.
- **Créer une tâche (\*)** : elle permet à l'utilisateur de créer une nouvelle tâche.
- **Paramètre (⚙)** : elle permet à l'utilisateur de changer le type d'un élément.
- **Monter (⬆)** : elle permet à l'utilisateur de monter un élément.
- **Descendre (⬇)** : elle permet à l'utilisateur de descendre un élément.
- **Supprimer (☒)** : elle permet à l'utilisateur de supprimer un élément.

### 4.3 Zone d'affichage de la liste

Cette zone est comme indiquée réservée à l'affichage de la liste en cours de création. On pourra donc voir l'arborescence de la liste, avec toutes ses informations. Cette zone sera organisée en une arborescence d'éléments. Pour cela, nous avons utilisé un TreeWidget.

Un élément est représenté avec un ??. Chaque élément possédera un nom (implémenter par une QLineEdit) suivi d'une date de fin (implémenter par une QDateEdit). Il possédera de plus une corbeille. A la fin de chaque élément, on aura une checkbox permettant à l'utilisateur de coché si oui ou non cette tâche à été effectuée. Plusieurs cas sont possible pour l'état de la checkbox :

- Dans le cas d'un élément d'une liste ordonnée, cette checkbox sera grisée si la tâche antérieure n'est pas cochée.
- Dans le cas où l'élément est une liste, tant que la checkbox de la liste est grisée, tous les éléments fils ont des checkbox grisées.
- Dans le cas d'une liste, sa checkbox sera automatiquement cochée lorsque tous les éléments de la liste seront eux aussi cochés.

Nous aurons une information supplémentaire au début de chaque élément. Dans le cas d'une liste non ordonnées, ses éléments sont représentés avec un tiret devant les informations de la liste, tandis que les éléments des listes ordonnées sont représenté à l'aide de numéro.

## 5 Le modèle

Afin de réaliser cette application nous avons du créer un modèle pour celle-ci. Voici le diagramme de classe de celui-ci : Le modèle de cette application est composé d'un pattern composite permettant d'implémenter une structure d'arbre et de composer les différents objets ensemble. Nous avons donc ici un *Component* qui est soit une *Task* (une tâche) ou une *List* (une liste) qui elle-même peut ensuite être une *ListOrdered* (une liste ordonnée), celle-ci hérite de la class *List*. Ce modèle permet donc comme expliqué ci-dessus de composer ces éléments et d'obtenir par exemple des listes de listes de tâches ...

Nous avons de plus une classe *History* permettant de stocker dans une liste circulaire les différents états de la liste en cours de création, après chaque action de l'utilisateur. Cette classe permettra l'utilisation des boutons *Annuler* et *Rétablir*.

## 6 Limite de l'application

## 7 Conclusion générale