

## RAPPORT DE PROJET

### Projet : Gestionnaire simple de tâches

#### **Taskinator Android**



COUTABLE Guillaume, RULLIER Noémie  
2 avril 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, <i>paper prototyping</i> et Scénarios</b>	<b>4</b>
3.1	Storyboard . . . . .	4
3.2	Scénarios . . . . .	4
3.2.1	Scénario 1 - Création/Utilisation/Suppression d'une liste . . . . .	4
3.2.2	Scénario 2 - Gestion des listes et sauvegarde . . . . .	4
<b>4</b>	<b>L'IHM</b>	<b>5</b>
4.0.3	Choix réalisés (Gestures, ...) . . . . .	5
4.1	Affichage des listes et tâches . . . . .	5
<b>5</b>	<b>Le modèle</b>	<b>6</b>
<b>6</b>	<b>Limite de l'application</b>	<b>7</b>
<b>7</b>	<b>Conclusion générale</b>	<b>8</b>

# 1 Introduction

L'objectif de ce projet fut de développer un gestionnaire simple 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 établi plusieurs étapes dans l'avancement du projet. Ce rapport présentera ces étapes les unes après les autres.

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

**Supprimer un élément :** cette fonctionnalité permet de supprimer une tâche ou une liste. 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 toutes ses tâches.

### 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 une liste ou une tâche. Dans le cas d'une liste, toutes ces tâches sont aussi montées/descendues d'un rang.

**Aperçu :** cette fonctionnalité permet à l'utilisateur d'avoir un aperçu de sa liste, se qui permettra l'affichage d'un nombre plus important de tâches.

## 3 Storyboard, *paper prototyping* 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 créé notre storyboard pour notre application :

FIGURE 1 – Le storyboard

### 3.2 Scénarios

Nous avons imaginé différents scénarios d'utilisation de notre application.

#### 3.2.1 Scénario 1 - Création/Utilisation/Suppression d'une liste

Ce premier scénario permet de créer une liste et d'y ajouter des tâches.

1. L'utilisateur crée une liste et lui donne un nom *Ski*.
2. Il crée ensuite une tâche dans cette liste et lui donne un nom *Bonnet*.
3. Il crée ensuite une autre tâche dans cette liste et lui donne un nom *Echarpe*.
4. Il souhaite maintenant échanger les tâches *Bonnet* et *Echarpe*, il reste longtemps appuyé sur la tâche *Bonnet* et la glisse un rang plus bas.
5. L'utilisateur supprime la tâche *Bonnet*.
6. L'utilisateur supprime la liste *Ski*. Une popup apparaît pour l'avertir que cette suppression supprimera aussi toutes les tâches de la liste.

#### 3.2.2 Scénario 2 - Gestion des listes et sauvegarde

Ce scénario permet de créer des listes et de les modifier. Il permet aussi de constater que l'état de l'application est sauvegardé.

1. L'utilisateur choisit de créer une liste et lui donne un nom *Course*.
2. Il décide ensuite de créer une nouvelle liste et lui donne un nom *Sac de voyage*.
3. Il décide ensuite d'inverser ces deux listes, pour cela il reste longtemps appuyé sur la liste *Sac de voyage* et la glisse un rang plus haut.
4. L'utilisateur va ensuite supprimer la liste *Course*.
5. L'utilisateur va ensuite ouvrir une autre application et revenir sur *Taskinator*. Il peut constater que l'application est ouverte avec l'état dans lequel on l'a quittée.
6. Cette fois-ci, l'utilisateur quitte l'application. De même s'il l'a lancée à nouveau, celle-ci est ouverte avec le dernier état dans lequel on l'a quittée.

## 4 L'IHM

### 4.0.3 Choix réalisés (Gestures, ...)

Dans cette section, nous expliquons les choix effectués et pourquoi nous avons choisi de mettre en place ces solutions :

- **Fonctionnalité de création de liste** : nous avons choisi de créer de nouvelles listes en cliquant sur le bouton "+" en bas de l'écran. Un nouvel item apparaît sur la page et possède le focus. L'utilisateur doit seulement donner le nom de cette nouvelle liste. Un nouveau bouton "+" apparaît dans cette liste, il permettra de créer de nouvelles tâches pour la liste.
- **Fonctionnalité de création de tâche** : nous avons choisi de créer de nouvelles tâches en cliquant sur le bouton "+" présent sous chaque liste pour ajouter une tâche à la liste correspondante. Lorsque l'utilisateur clique sur ce bouton, un nouvel item de tâche apparaît dans la liste (au-dessus du bouton "+") avec le focus. L'utilisateur doit seulement renseigner le nom de la tâche. Nous avons choisi de procéder ainsi plutôt que de mettre un bouton de création dans l'entête de la liste, pour éviter à l'utilisateur de remonter en haut de la liste à chaque fois qu'il souhaite ajouter une tâche (cela peut vite devenir contraignant).
- **Fonctionnalité Monter et Descendre** : nous avons décidé d'accéder à cette fonctionnalité seulement grâce aux gestures. En effet, il suffit à l'utilisateur de rester longtemps appuyer sur une liste ou tâche et de la glisser à l'endroit voulu. Attention, lors de l'utilisation de cette fonctionnalité dans une liste ordonnée ; si on monte ou descend un élément de tel façon qu'un élément coché se trouve être en dessous d'un élément non coché dans la liste, alors le déplacement ne sera pas activé. (Cette dernière précaution n'a pas été implémentée)
- **Fonctionnalité Supprimer** : nous avons choisi d'ajouter cette possibilité de suppression sur chaque élément pour que l'utilisateur puisse supprimer plus rapidement. De plus nous avons décidé de faire apparaître une fenêtre d'avertissement car la suppression d'une liste entraîne la suppression de toutes ses tâches. Pour plus de sécurité, le focus de la validation de la suppression est par défaut sur *Non*.

L'application est tout simplement composée d'une zone réservée à l'affichage de la liste en cours de création et d'une bar d'action.

FIGURE 2 – L'application

### 4.1 Affichage des listes et tâches

Afin d'afficher l'ensemble des listes créées et de leur(s) tâche(s), on a décidé d'utiliser `ExpandableListView`, permettant d'afficher l'ensemble des listes et chacune d'entre elle sa ou ses tâche(s). La liste est décrite à l'aide du xml `list.xml` et est représentée par un `EditText` permettant de donner/modifier le nom de celle-ci et d'un `ImageButton` permettant d'ajouter la fonctionnalité de suppression de liste. La tâche est décrite par le fichier `task.xml` et est représentée par une `checkbox`, un `EditText` permettant de taper le nom de la tâche et d'un `ImageButton` permettant d'ajouter la fonctionnalité de suppression sur la tâche.

## 5 Le modèle

Afin de réaliser cette application nous nous sommes basés sur le modèle suivant dont voici le diagramme de classes :

FIGURE 3 – Diagramme de classe du modèle

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 peut être, soit une *Task* (une tâche), soit une *List* (une liste) qui elle-même peut ensuite être une *SortedList* (une liste ordonnée), celle-ci hérite de la classe *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 pouvons donc gérer tous ces composants au sein de ce modèle et ainsi, ajouter un composant dans une liste, le supprimer, le déplacer d'un rang (dans les deux sens). Nous disposons aussi de toutes les fonctions permettant de déterminer si un composant est cochable ou non.

## 6 Limite de l'application

Notre application à ses propres limites. On pourrait imaginer d'ajouter certaines fonctionnalités comme par exemple :

- Recherche ( 🔍 ) : qui permettrait à l'utilisateur d'afficher seulement les listes qui ont un nom ou qui ont des tâches dont le nom contient la chaîne du filtre.
- Partage ( 📎 ) : qui permettrait à l'utilisateur de partager sa ou ses listes sur des réseaux sociaux ou par mail.
- Favoris ( ❤️ ) : qui permettrait à l'utilisateur de mettre certaines listes ou tâches en favoris et d'y avoir accès via un onglet favoris. Cela permettrait d'avoir un accès plus rapide.



## 7 Conclusion générale

Ce projet nous a permis de réfléchir à la structure de l'IHM afin que celle-ci soit la plus intuitive possible pour l'utilisateur. Réfléchir à la position des boutons, le nom ou l'icône les représentant, comment l'ajout des composants se fera au sein de la liste, la gesture à utiliser, l'utilisation d'un menu ... Et donc, d'orienter notre conception sur l'IHM, plus que sur le modèle, et ainsi expérimenter une autre approche de conception.