Projet d’Algorithmique et Structure de données

Sujet n°6 : **Historique d’action dans un éditeur de texte** (Undo/Redo)

**Membres du groupe :**

* TSAFACK NGOUFACK Ernis Merkel
* NOUBISSI MEIKEM Indira Leslie
* NANA Laurianne Audrey
* TANKAM SOAH Raoul Daril

**Chargé** **de** **cours** : Ing. ATANGANA Guy Martial

**Plan**

1. **Introduction**
   1. But du document
   2. Portée du projet
   3. Définitions, Acronymes et Abréviations
   4. Références
   5. Vue d’ensemble du reste du document
2. **Description Générale** 
   1. Contexte du produit
   2. Besoins des utilisateurs
   3. Caractéristiques du produit
   4. Contraintes
   5. Hypothèses et Dépendances
3. **Exigences spécifiques** 
   1. Exigences fonctionnelles
   2. Exigences non fonctionnelles
   3. Exigences d’Interface Externe
4. **Préparation à la livraison** 
   1. Plan de déploiement
   2. Exigences de formation
   3. Exigences de Documentation
   4. Exigences de Maintenance
5. **Annexes**

**1.Introduction**

* 1. **But du document**

Ce document est destiné à toute personne souhaitant réaliser un éditeur de texte doté d’un historique d’action de A à Z que vous soyez débutant ou aguerri. Il est également fait pour présenter le résultat de notre travail et les méthodes employés pour réaliser notre éditeur de texte et y intégrer un historique d’action afin de permettre à nos camarades de pouvoir reproduire ce projet de façon ludique et efficace.

* 1. **Portée du projet**

Ce projet permet à la fois d’implémenter les fonctions ‘’Annuler’’ et ‘’Rétablir’’ dans un éditeur de texte. Chaque fois qu’un utilisateur effectue une action (ajouter un caractère, supprimer, copier, coller, changer la couleur ou la taille du texte) celle-ci est stockée dans une pile d’actions. Lorsqu’il appuie sur ‘’Undo’’, l’action est dépilée de la pile des actions et poussée dans une autre pile ‘’Redo’’. Le système permet de revenir en arrière dans l’historique d’édition, puis de réappliquer les actions annulées si besoin.

* 1. **Définitions, Acronymes et Abréviations**
* Une pile est une structure de données basée sur le principe <<Dernier arrivé, Premier sorti>>, ou LIFO (Last In, First Out), ce qui veut dire que les derniers éléments ajoutés à la pile seront les premiers à être récupérés. Le fait de dépiler consiste donc à prendre ce dernier élément pour l’utiliser dans notre programme.
* Les fonctions Undo (Annuler) et Redo (Rétablir) permettent à l’utilisateur d’annuler ou de de rétablir la ou les dernières actions qu’il a effectuées, généralement sur un document, afin de le faire revenir à un état antérieur.
* Qt est un Framework C++ pour interface graphique qui permet d’utiliser le principe d’abstraction pour écrire du code sans pour autant connaitre toutes les ficelles du langage UI (Interface Utilisateur).
* Le principe d’abstraction est un concept qui consiste à cacher les détails de fonctionnement d’un système à l’utilisateur pour lui fournir uniquement le nécessaire.
  1. **Références**

Qt documentation, spécifications IEEE 830

Référence widget : <https://www.flaticon.com>.

* 1. **Vue d’ensemble du reste du document**

Dans la suite du document, nous vous présenterons les différentes raisons qui nous ont emmenés à développer ce projet, les différents éléments de notre programme, les contraintes auxquelles nous avons été confrontés, les différentes exigences de ce projet, notre plan de déploiement et les différentes améliorations que nous pourrions faire dans le futur.

**2.Description Générale**

**2.1 Contexte du produit**

Ce projet est fondamental pour comprendre comment conserver et restaurer l’état d’une application. Le fait de pouvoir utiliser les fonctions Undo et Redo permettent d’optimiser l’utilisation d’un éditeur de texte afin de rendre la saisie beaucoup plus fluide. Par exemple, on aura plus besoin de réécrire un texte que l’on avait au préalable effacer il suffira simplement de le rétablir puisqu’il sera stocké dans une autre pile. Notre produit sera développé grâce à la bibliothèque Qt qui nous fournira une interface graphique (sans base de données). Ce logiciel peut donc remplacer un éditeur de texte basique étant fait via un Framework, il est compatible avec tout environnement de développement où Qt est installé et fonctionnel.

**2.2 Besoin des utilisateurs et objectifs du projet**

Ce projet vise à faciliter la saisie et la mise en forme de texte pour tous nos utilisateurs, les différents éléments qui sont intégrés dans notre éditeur de texte (supprimer, ajouter des caractères, copier, colle, changer la couleur ou la taille du texte, Annuler, Rétablir) permettent de rendre la saisie de texte beaucoup plus rapide et optimale. Elle permet de donner une utilisation simple et efficace des piles dans un programme en manipulant les fonctions d’empilement et de désempilement (‘’Undo’’ et ‘’Redo’’).

**2.3 Caractéristiques du produit**

Les fonctionnalités de notre éditeur de texte sont : l’édition optimisée de texte, l’ouverture et la sauvegarde de fichiers .txt, les fonctions Copier/Couper/Coller, Undo/Redo (Annuler/Rétablir) pour une gestion plus rapide du texte. Ainsi que des options de rendu visuel pour modifier la couleur et la taille du texte.

**2.4 Contraintes**

Les contraintes liées sont notamment les contraintes réglementaires : Qt Creator impose des droits d’auteurs sur tout projet fait avec dans le cadre de la commercialisation. Le projet prend en pratique quelques heures pour être fait et nécessite juste d’avoir un PC pour pouvoir coder peut importe le modèle que vous avez (que ce soit un ordinateur bas de gamme ou haut de gamme, il n’y a pas d’importance), il ne nécessite pas beaucoup de place dans votre machine. Il n’est pas très sécurisé car toute personne ayant accès à votre PC peut voir toutes les modifications apportées à votre document (tant que vous laissé le document ouvert après avoir fait ces modifications).

**2.5 Hypothèses et Dépendances**

* L’utilisateur a installé Qt et ce dernier est fonctionnel sur son PC.
* L’utilisateur maitrise les fonctions de base d’un éditeur de texte et peut se servir correctement de chacune d’entre elle.

**3. Exigences Spécifiques**

**3.1** **Exigences fonctionnelles (ERF)**

* ERF001 : Ouvrir un fichier texte (.txt)
* ERF002 : Sauvegarder des modifications
* ERF003 : copier, couper, coller
* ERF004 : Modifier la couleur du texte
* ERF005 : Modifier la taille de la police
* ERF006 : Afficher une fenêtre de configuration

**3.2 Exigences non fonctionnelles (ERNF)**

* ERNF001 : Interface simple et intuitive
* ERNF002 : Temps de chargement < 2s
* ERNF003 : L’application doit être stable et ne pas planter

**3.3 Exigences d’Interface Externe**

* UI : Interface Utilisateur en Qt Creator
* Logiciel : Interaction avec le système de fichiers local uniquement.

**4. Préparation à la livraison**

**4.1 Plan de déploiement**

L’application sera fournie sous forme exécutable Windows.

**4.2 Exigences de Formation**

Ce projet ne demande que d’avoir des bases en C++, si vous maitriser déjà l’implémentation de pile dans un programme ainsi que les fonctions principales liés à celle-ci (insérer un élément dans une pile, le supprimer, le déplacer, vérifier si la pile est vide) vous pouvez déjà réaliser ce projet tout seul (vous pouvez également vous aider de Qt).

En ce qui concerne l’utilisateur, il n’a aucun besoin spécifique de formation.

**4.3** **Exigences de Documentation**

L’utilisation de notre éditeur de texte est très intuitive et ne nécessite pas de mode d’emploi. Le principe étant assez simple et similaire au précèdent éditeur que vous avez déjà utilisé.

**4.4 Exigences de maintenance**

Evolutivité prévue pour ajouter la gestion de nouveaux formats (JPEG, PDF).

**5.Annexes**

* **Glossaire**
* UI : Interface Utilisateur
* Qt : Framework graphique
* ERF : Exigence Fonctionnelle
* ERNF : Exigence Non Fonctionnelle
* JPEG : Joint Photographic Expert Group (format d’image)
* PDF : Portable Document Format (format de fichier)
* **Prototype**

Voir l’interface dans Qt Creator

* **Historique**

Version initiale du projet réalisé dans le cadre d’un projet étudiant.