Programmation Système

Livrable 1

Cahier des charges v1.0 et architecture logicielle

EasySave de chez ProSoft

DESPREZ Julien, JAGUENEAU Louis,

LORIT Nathan, PESCHEL Paul

09/02/2024

Table des matières

Table des figures

Introduction

L'objectif principal est de fournir une application console en utilisant .NET Core pour gérer un ensemble de fonctionnalités concernant une gestion limitée de sauvegarde de données, ainsi qu'un accès aux fichiers de logs.

Ce projet est disponible sur la plate-forme de dépôt GitHub à l'adresse suivante : EasySave. Il est possible de soit le télécharger directement au format .zip, soit de cloner un projet avec le lien URL depuis le logiciel Visual Studio 2022. Pour ce faire, il est nécessaire d'installer les bibliothèques recommandées. De plus, si l'option pour le téléchargement au format .zip est choisie, il suffit d'ouvrir un projet ou une solution depuis Visual Studio 2022.

Rappel des besoins

- Le logiciel est une application console utilisant le .NET Core.
- Le logiciel doit permettre de créer jusqu'à cinq travaux de sauvegarde.
- Un travail de sauvegarde est défini par :
    - Un nom de sauvegarde.
    - Un répertoire source.
    - Un répertoire cible.
    - Un type (complet, différentiel).
- Le logiciel doit être utilisable à minima par des utilisateurs anglophones et francophones.
- L'utilisateur peut demander l'exécution d'un des travaux de sauvegarde ou l'exécution séquentielle de l'ensemble des travaux.
- Le programme peut être lancé par une ligne de commande :
    - Exemple 1 : « 1-3 » pour exécuter automatiquement les sauvegardes 1 à 3.
    - Exemple 2 : « 1 ;3 » pour exécuter automatiquement les sauvegardes 1 et 3.
- Les répertoires (sources et cibles) pourront être sur :
    - Des disques locaux.
    - Des disques externes.
    - Des lecteurs réseaux.
- Tous les éléments du répertoire source (fichiers et sous-répertoires) doivent être sauvegardés.
- Fichier de logs journalier :

Le logiciel doit écrire en temps réel dans un fichier log journalier l'historique des actions des travaux de sauvegarde. Les informations minimales attendues sont :

    - Horodatage.
    - Nom de sauvegarde.
    - Adresse complète du fichier source (format UNC).
    - Adresse complète du fichier de destination (format UNC).
    - Taille du fichier.
    - Temps de transfert en ms (négatif si erreur).
- Fichier d'états en temps réel :

Le logiciel doit enregistrer en temps réel, dans un fichier unique, l'état d'avancement des travaux de sauvegarde. Les informations à enregistrer pour chaque travail de sauvegarde sont :

    - Appellation du travail de sauvegarde.
    - Horodatage.
    - État du travail de sauvegarde comme actif ou non actif.

Si le travail est actif :

- Le nombre total de fichiers éligibles.
- La taille des fichiers à transférer.
- La progression :
  - Nombre de fichiers restants.
  - Taille des fichiers restants.
  - Adresse complète du fichier source en cours de sauvegarde.
  - Adresse complète du fichier de destination.

- Les emplacements des deux fichiers (logs journaliers et états en temps réel) devront être étudiés pour fonctionner sur les serveurs des clients. De ce fait, les emplacements du type « C:\temp\ » sont à proscrire.
- Ces fichiers et les éventuels fichiers de configuration seront au format JSON. Pour permettre une lecture rapide via Notepad, il est nécessaire de mettre des retours à la ligne entre les éléments JSON. Une pagination serait un plus.

Remarque importante : Si le logiciel donne satisfaction, la direction vous demandera de développer une version 2.0 utilisant une interface graphique WPF (basée sur l'architecture MVVM).

Diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation que nous avons élaboré est un outil fondamental pour décrire les interactions entre les acteurs (utilisateurs ou systèmes externes) et le système en question. Son objectif principal est de représenter les différentes fonctionnalités offertes par le système du point de vue de l'utilisateur.

Figure 1: Diagramme de cas utilisation

Diagrammes d’activité

Le diagramme d'activité que nous avons conçu est un outil utilisé pour modéliser les flux de contrôle et les comportements des processus ou des systèmes logiciels. Son objectif principal est de représenter graphiquement les différentes activités d'un processus, ainsi que les transitions entre ces activités.

Ce diagramme est particulièrement utile pour plusieurs raisons. Tout d'abord, il permet de visualiser de manière claire les étapes d'un processus, ainsi que les décisions prises à chaque étape. Cela facilite la compréhension du fonctionnement du processus et permet d'identifier les éventuels goulets d'étranglement ou les points d'amélioration.

De plus, le diagramme d'activité est un outil utile pour la documentation et la communication des processus. Il offre une représentation visuelle intuitive des étapes à suivre, ce qui facilite la transmission des connaissances et la formation des utilisateurs.

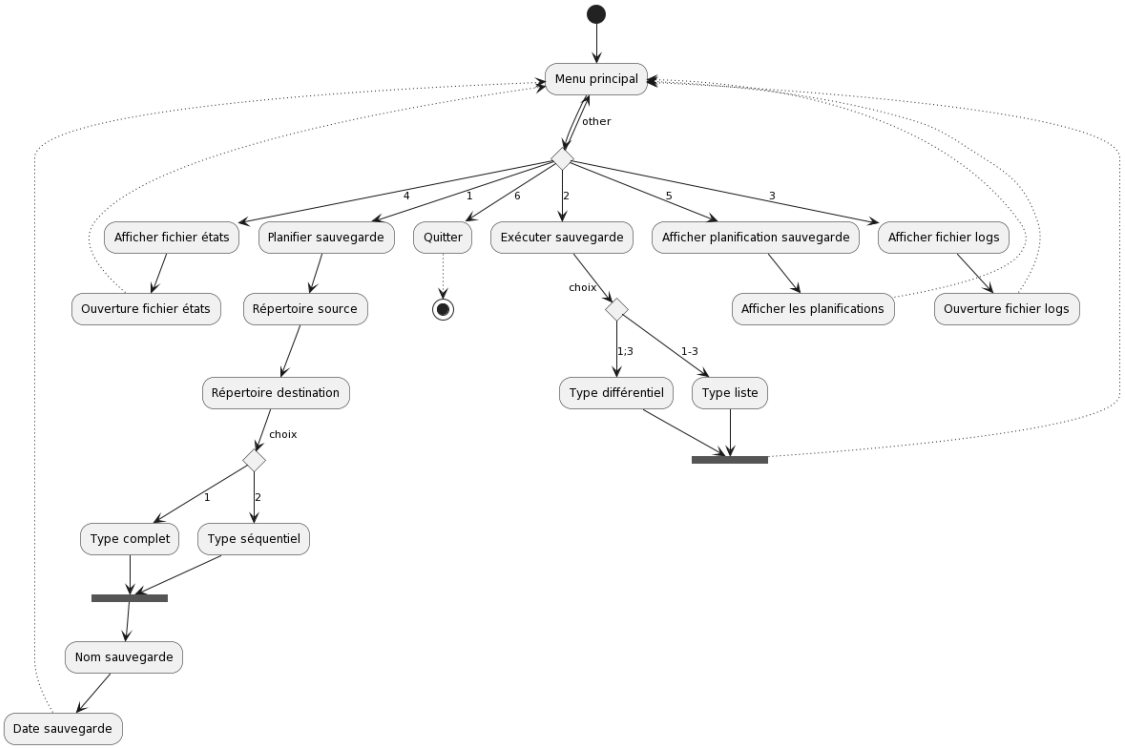

Figure 2 : Diagramme d'activité

Diagrammes de classes

Le diagramme de classe que nous avons réalisé est un outil essentiel pour représenter la structure statique d'un système logiciel. Son but principal est de visualiser les différentes classes du système, ainsi que leurs relations et attributs.

Ce diagramme offre plusieurs avantages. Tout d'abord, il permet une meilleure compréhension de l'architecture du logiciel en identifiant clairement les entités et leurs interactions. Cela facilite la communication entre les membres de l'équipe de développement et favorise une conception plus cohérente et modulaire du système.

En outre, le diagramme de classe joue un rôle dans la phase de conception du logiciel. Il aide les développeurs à organiser et à structurer leur code de manière efficace, en définissant les différentes classes, leurs attributs et méthodes, ainsi que les relations. Cela permet de garantir une meilleure évolutivité, maintenabilité et réutilisabilité du code.

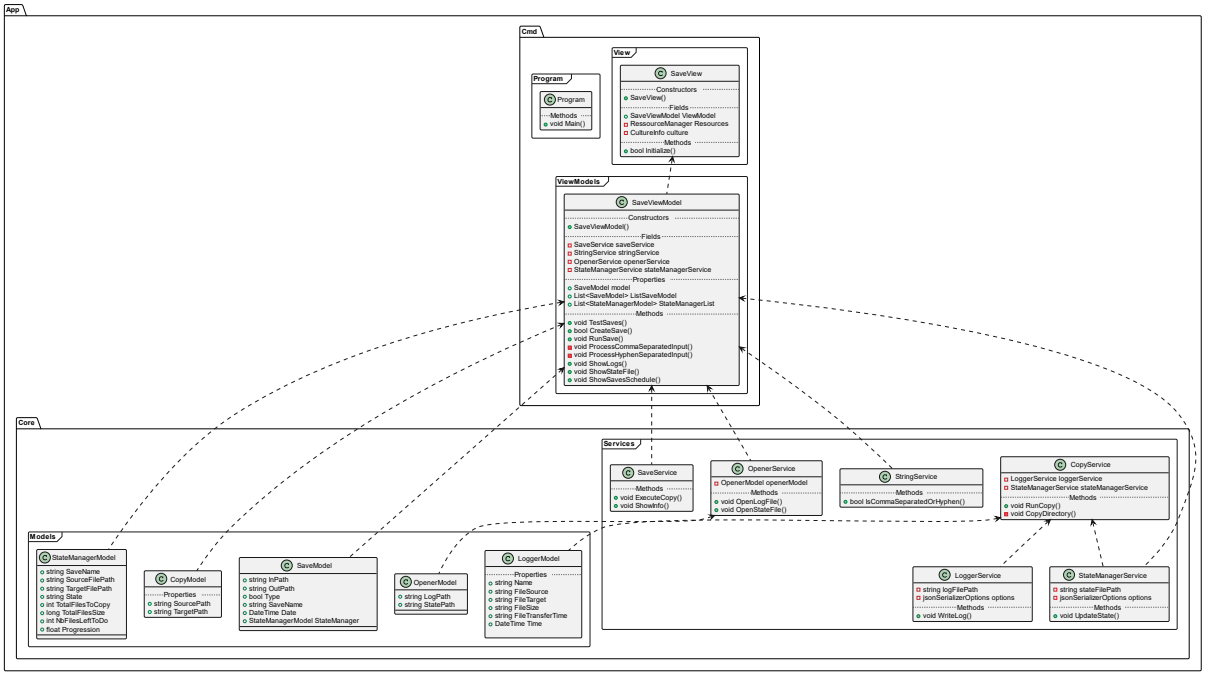

Figure 3 : Diagramme de classes

Diagrammes de séquence

Le diagramme de séquence que nous avons élaboré est un outil utilisé pour modéliser et visualiser les interactions entre les différents objets d'un système logiciel au cours du temps. Son principal objectif est de décrire comment les objets interagissent les uns avec les autres pour réaliser un scénario spécifique ou exécuter une fonctionnalité particulière de l'application.

Ce diagramme revêt une importance dans le processus de conception et de développement logiciel pour plusieurs raisons. Tout d'abord, il permet de capturer de manière précise et détaillée le comportement dynamique du système, en mettant en évidence l'ordre des messages échangés entre les objets et les actions qui en découlent. Cela aide les développeurs à mieux comprendre le flux d'exécution du système et à identifier les éventuels problèmes ou points d'amélioration à prendre en compte.

Enfin, ce type de diagramme est également utile pour valider et tester le système, en permettant aux développeurs de simuler différents scénarios d'utilisation et de vérifier si le système réagit de manière appropriée.

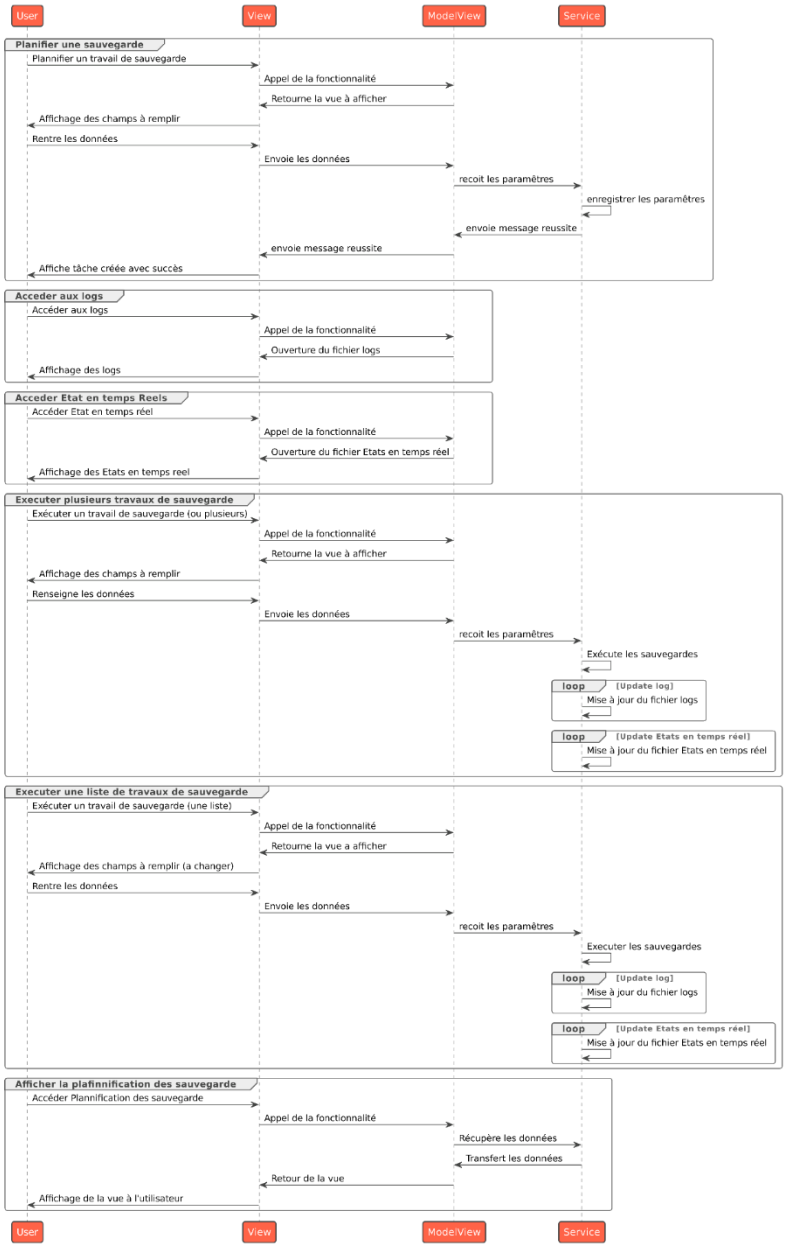

Figure 4 : Diagramme de séquence

Conclusion

La mise en œuvre de ce projet de développement d'une application console avec .NET Core pour la gestion de sauvegardes de données a été menée avec succès. À travers l'introduction d'une interface utilisateur conviviale et efficace, notre objectif était de répondre aux besoins essentiels des utilisateurs francophones et anglophones tout en assurant une gestion robuste des sauvegardes.

Ce projet, hébergé sur GitHub sous le nom d'EasySave, offre une solution flexible pour la création et l'exécution de travaux de sauvegarde, avec une prise en charge des répertoires locaux, externes et réseau. Grâce à des commandes simples et intuitives, l'utilisateur peut déclencher des sauvegardes individuelles ou séquentielles selon ses besoins spécifiques.

L'ajout de fonctionnalités telles que la génération de fichiers de logs journaliers et d'états en temps réel assure une traçabilité complète des actions effectuées et l'observation en temps réel de l'avancement des sauvegardes. En garantissant la conformité avec les exigences de formatage JSON pour la facilité de lecture et la pagination, nous avons optimisé la lisibilité des fichiers générés.

L’objectif maintenant est de développé une version 2.0 de la même application en ajoutant une interface graphique WPF basée sur l'architecture MVVM. Cette évolution permettra une expérience utilisateur enrichie et une gestion encore plus intuitive des sauvegardes.