Projet Programmation Système - Livrable 1

EasySave version 1.0

31/01/2022

CESI Ecole d'Ingénieurs

Saint-Nazaire

Anne-Laure GAUDON : Tuteur pédagogique

FISA A3 Informatique - Groupe 2

COUTEAUX Corentin

DEMBELE Romaric

RAMBIER Ewen

Mode de diffusion :
Libre

TABLE DES MATIERES

INTRODUCTION

L’avènement du numérique dans nos sociétés à bouleverser la façon de travailler. Les données sont de plus en plus sauvegardées et cela grâce à des logiciels qui offrent cette fonctionnalité. C’est dans ce contexte que se place EasySave, un logiciel de sauvegarde de données qui sont sous forme de fichiers informatiques. La réalisation de cette application requiert une modélisation logicielle préalable, un certain nombre d’outils et de technologies informatiques. Aussi, un manuel d’utilisation du logiciel devra permettre une utilisation plus facile et efficiente de cette dernière.

I. DIAGRAMMES UML

Pour avoir un aperçu visuel d'un logiciel et de ses fonctionnalités du point de vue d'un développeur et d'un non-développeur, un certain nombre de diagrammes normalisés existent. Le langage de modélisation unifié (en anglais, UML – Unified Modeling Language) permet de réaliser ces diagrammes. Pour notre logiciel, une présentation des 4 de ces diagrammes sera effectué.

I.1 Diagramme de cas d'utilisation

Figure 1 : Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation permet d'avoir un aperçu des différentes fonctionnalités du logiciel. C'est un diagramme qui pourra être présenté à des acteurs du projet qui n'ont pas forcément un bagage technique.

Ainsi, nous pouvons voir sur l'image ci-dessus les fonctionnalités principales du logiciel à savoir : l'exécution d'un travail de sauvegarde, l'exécution séquentielle de tous les travaux de sauvegarde, la création, le renommage et la suppression d'un travail de sauvegarde ainsi que le changement de langue du logiciel.

I.2 Diagramme d'activité

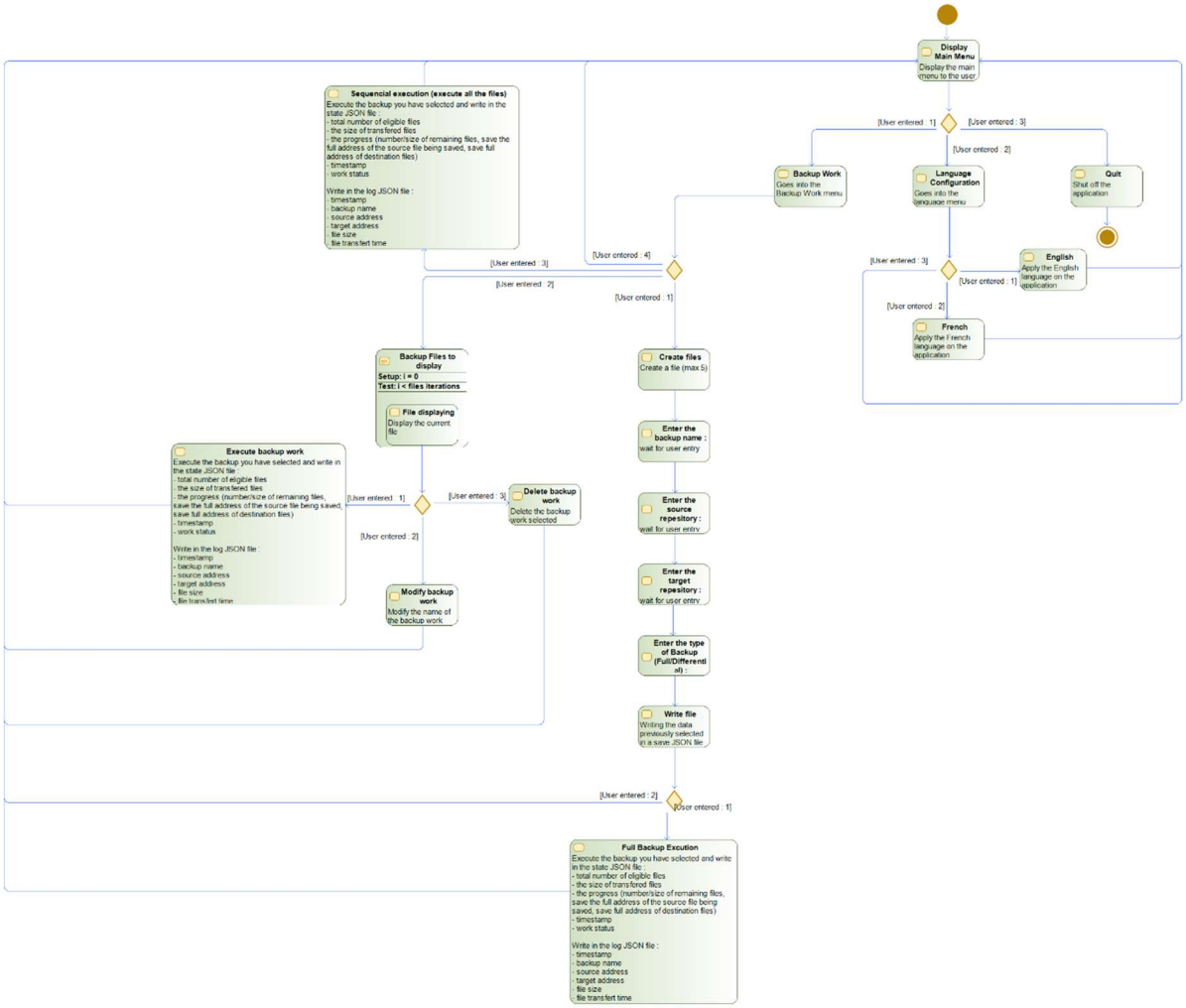

Figure 2 : Diagramme d'activité

Un diagramme d'activité est un schéma montrant le déclenchement d'évènement en fonction des états du système. Il permet d'exprimer une dimension temporelle des cas d'utilisations du logiciel. Représenter un cas d'utilisation d'un logiciel par un diagramme d'activité correspond à traduire algorithmiquement le cas d'utilisation.

Ainsi, nous pouvons à partir du diagramme ci-dessus (Figure 2) voir que depuis le menu principal de l'application, l'utilisateur a le choix entre quitter l'application (l'utilisateur tape le chiffre 3 sur son clavier), configurer la langue du logiciel (l'utilisateur tape le chiffre 2 sur son clavier), et réaliser un travail de sauvegarde (l'utilisateur tape le chiffre 1 sur son clavier). En fonction de son choix, plusieurs autres choix se présentent à lui (se référer au diagramme pour voir toutes les possibilités).

I.3 Diagramme de classes

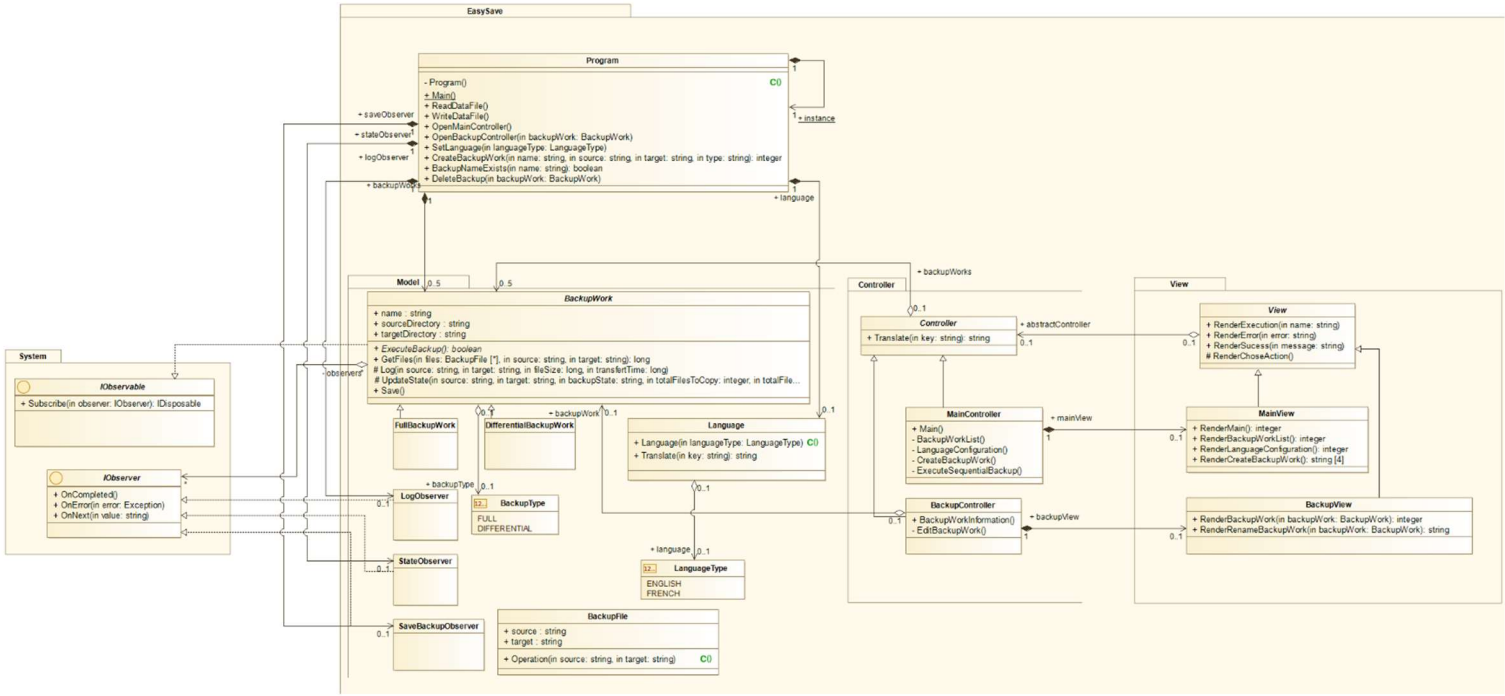

Figure 3 ; Diagramme de classes

Afin de disposer de la structure du logiciel à implémenter, le diagramme de classes intervient en montrant les classes et les interfaces du programme ainsi que les relations entre les différentes classes qui peuvent être de type association, agrégation, composition et héritage.

Le diagramme de classes de notre logiciel présente les classes en les regroupant en fonction de leur nature dans l'architecture MVC (Model View Controller). L'architecture du logiciel se présente sous cinq grandes parties à savoir la partie Modèle, Vue, Contrôleur, Program et Système.

I.4 Diagrammes de séquence

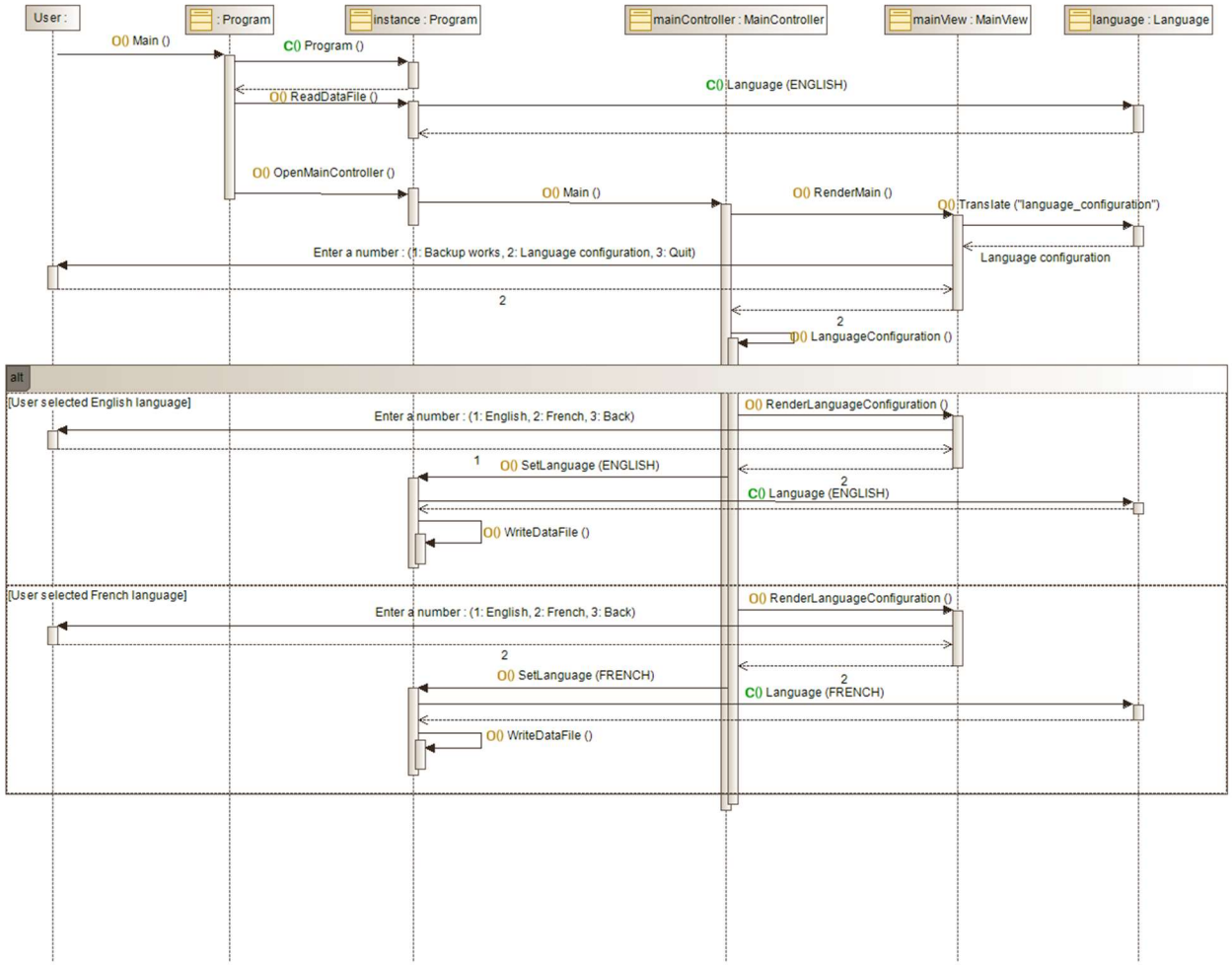

Figure 4 : Diagramme de séquence – choix de la langue

Le diagramme de séquence est un schéma illustrant les interactions chronologiques entre les objets d'un système (dans notre cas le logiciel de sauvegarde) pour un cas d'utilisation donné du logiciel.

I.4.1 Choix de la langue

La figure 4 illustre, à partir du diagramme de séquence, les interactions nécessaires entre les objets du logiciel dans l'optique d'effectuer un changement de langue (deux options de langue possible – l'anglais et le français).

I.4.2 Exécution séquentielle

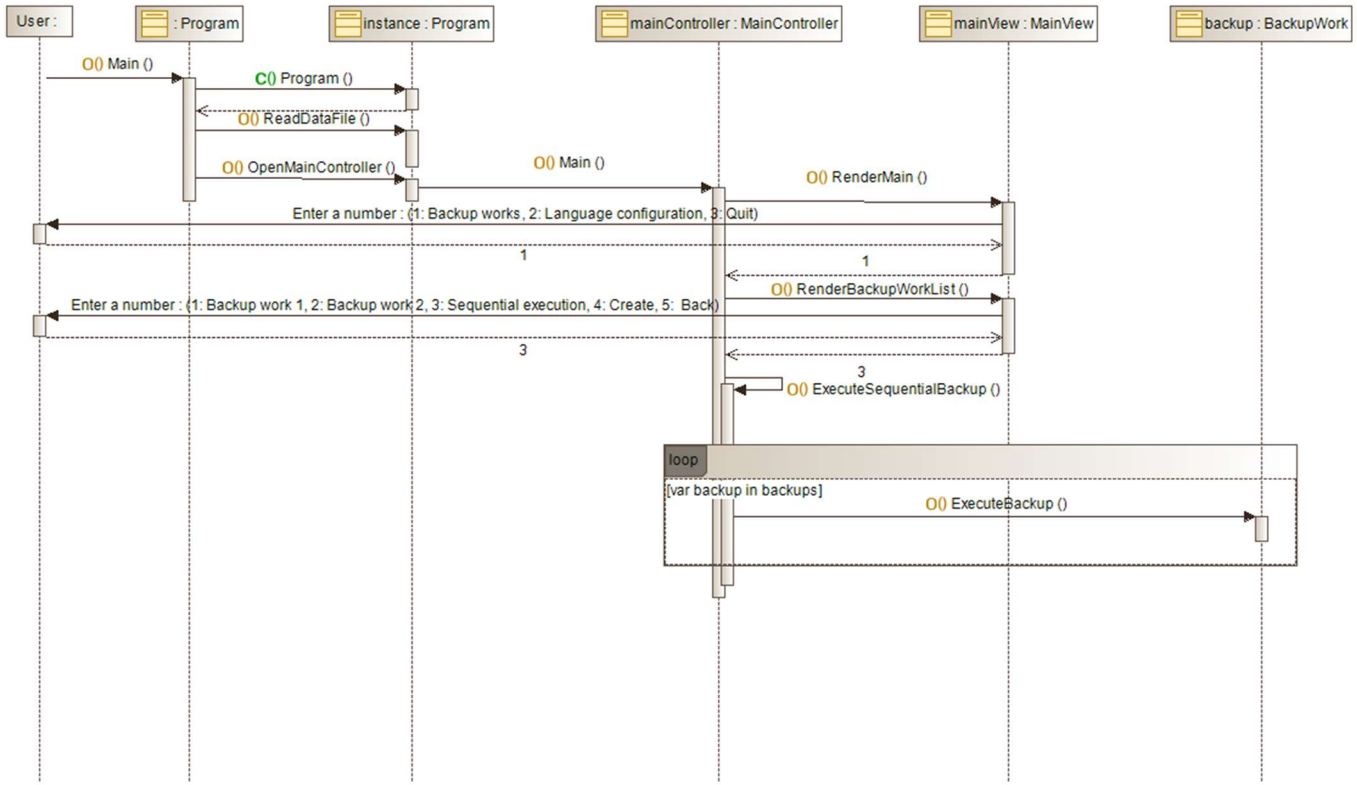

Figure 5 : Diagramme de séquence – Exécution séquentielle

La figure 5 illustre les interactions nécessaires entre l'utilisateur et les objets du logiciel afin de réaliser un travail de sauvegarde séquentiel.

I.5 Diagramme de composants

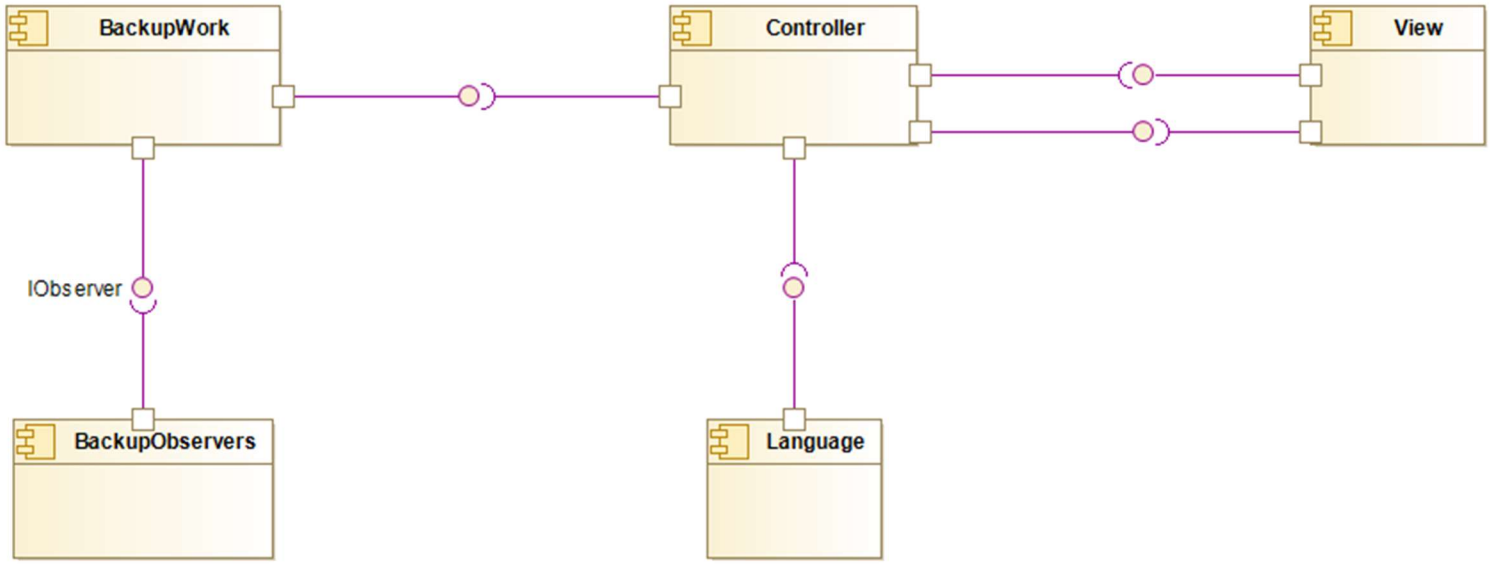

Figure 6 : Diagramme de composants

Le diagramme de composants est un schéma illustrant une vue globale des composants du système ainsi que les dépendances entre ces composants. Les composants peuvent être des classes, des ensembles de classe, des paquets, des bibliothèques logiciels, etc.

Notre diagramme (figure 6) montre les composants (BackupWork, Controller, View BackupObservers, Language) ainsi que leurs dépendances mutuelles.

II. OUTILS ET TECHNOLOGIES UTILISÉS

II.1 Patrons de conception (Design patterns)

Afin de faciliter la conception logicielle, les patrons de conception ayant déjà fait leur preuve auprès des développeurs de logiciels, entrent en jeu. Il existe une multitude de patrons de conception ; chacun solutionnant un problème spécifique donné. Selon Wikipédia, un patron de conception est un arrangement caractéristique de modules, reconnu comme bonne pratique en réponse à un problème de conception d'un logiciel.

En ce qui concerne la version 1 de notre logiciel de sauvegarde, deux patrons de conception à savoir l'architecture **MVC** (Modèle -Vue -Controller) ainsi que le patron « **Observateur »** ont été implémentés.

L'architecture MVC est un motif d'architecture segmentée en trois couches à savoir :

- Un modèle contenant les données à afficher.
- Une vue contenant la présentation de l'interface du logiciel.
- Et un contrôleur correspond à la logique des actions de l'utilisateur.

Le patron de conception « Observateur » est utilisé pour envoyer des signaux à des modules du logiciel appelés observateurs afin qu'ils entreprennent des actions en fonction de ces signaux (notifications). Dans le cas de notre logiciel de sauvegarde, il est utilisé pour la mise à jour les logs.

II.2 DevOps

II.2.1 Azur DevOps GIT

Afin de disposer d'un versionnage de notre logiciel de sauvegarde durant la conception ainsi que permettre un travail collaboratif nous avons utiliser GIT qui est un logiciel de gestion de versions décentralisé.

II.2.2 GitHub

La gestion de développement à distance de notre logiciel s'est réalisée avec GitHub.

II.3 Plateforme de développement

Le développement de la première version de notre logiciel a été effectué sur Visual Studio 2019 avec le Framework .Net Core 3.X

II.4 Package NuGet Newtonsoft.Json

Afin de convertir un objet .Net en objet Json, nous utilisons la package NuGet **Newtonsoft.Json** dans notre logiciel.

III. DOCUMENTATION UTILISATEUR

Le logiciel offre les fonctionnalités suivantes à un utilisateur :

```
1. Backup works
2. Language configuration
3. Quit
Choose an action : 2
1. English
2. Français
3. Back
Choose an action : 2
Langue définit en Français
1. Travaux de sauvegarde
2. Configuration de la langue
3. Quitter
```

Figure 7 : Changement de langue

Changement de langue

Au démarrage du logiciel, l'utilisateur a la possibilité de changer de langue en tapant sur le chiffre 2, puis sur la touche Entrer sur son clavier et ensuite choisir sa langue (Figure 7).

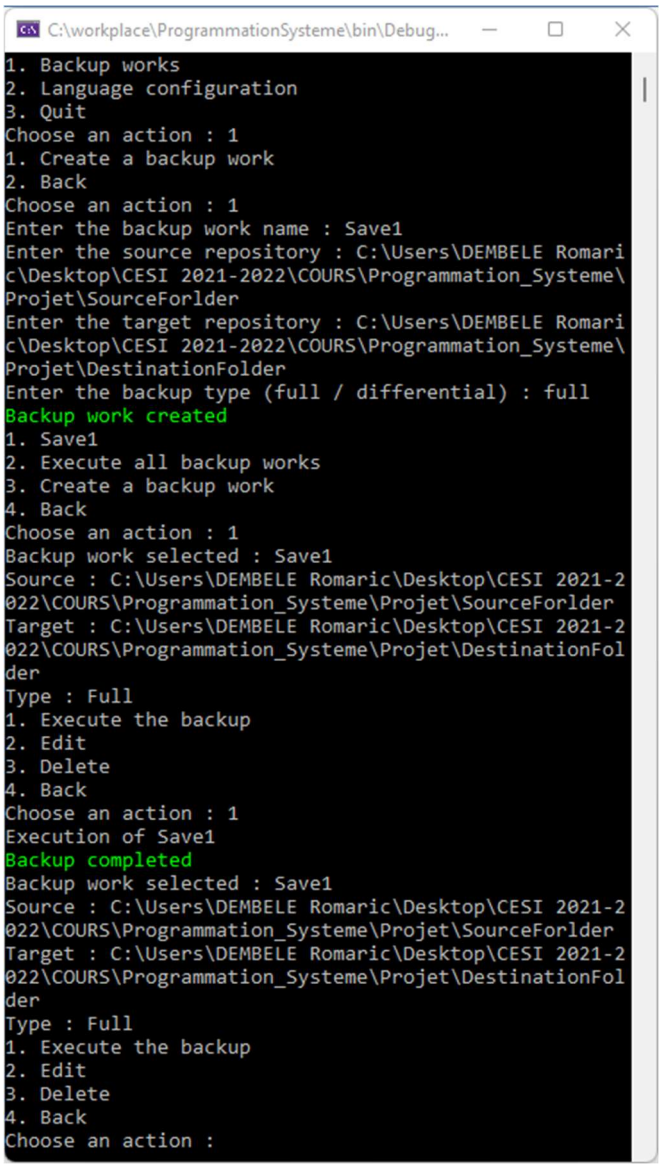

Figure 8 : Création et exécution d'un travail de sauvegarde

Création, Exécution, Suppression d'un travail de sauvegarde

Après avoir créé un travail de sauvegarde (lors de la création, le type de sauvegarde pourra être choisi – Full ou différentielle), l'utilisateur pourra décider l'exécuter, de l'éditer ou de supprimer en suivant les différentes options proposées sur l'écran (Voir Figure 8 ci-contre).

Exécution de tous les travaux de sauvegarde

L'utilisateur pourra également décider d'exécuter tous les travaux de sauvegarde qu'il a préalablement créé s'il le souhaite

(L'option se présente s'il a déjà créé au moins un travail de sauvegarde).

CONCLUSION

En guise de conclusion, à travers les diagrammes de cas d'utilisation, d'activité, de classes, de séquence et de composants, la modélisation du logiciel de sauvegarde EasySave a permis une implémentation plus aisée, prenant en compte toutes les spécifications du cahier de charges. En ce qui concerne la gestion des versions du logiciel et le travail collaboratif, les outils GIT et GitHub ont été d'un grand atout.

La segmentation du code du logiciel a permis de bénéficier d'une modularité et d'une évolutivité du logiciel. D'ailleurs, la deuxième version du logiciel qui possèdera cette fois une interface graphique s'appuiera sur cette modularité pour une implémentation plus rapide et aisé.