PROGRAMMATION SYSTEME

LIVRABLE 1

BRIAND Lucile
KHADHAR Dany
DIAS Bruno
MONAQUE Victor

TABLE DES MATIERES

INTRODUCTION

Dans ce projet *Programmation Système*, nous incarnons une société de développement d'un logiciel de sauvegarde. Ce logiciel, nommé EasySave, nous a été commandé avec un cahier des charges respectant certains critères.

Le logiciel EasySave est une application facilitant la sauvegarde de fichiers. Pour cela, elle doit être capable de créer et des travaux de sauvegarde, d'en supprimer, et de les exécuter. L'application doit permettre de suivre le déroulement des exécutions, et de copier les fichiers souhaités de manière complète ou bien différentielle. Enfin, ce logiciel doit pouvoir être utilisé par des utilisateurs anglophones et francophones au minimum. Le développement de notre projet se fera entièrement en C# et .NET Core, ainsi qu'à l'aide de l'éditeur Visual Studio 2019.

Dans ce premier livrable, nous devions concevoir et développer une première version du logiciel sous la forme d'une application console. Nous verrons donc dans un premier temps comment nous avons modélisé notre application sous forme de plusieurs diagrammes UML. Enfin, nous expliquerons rapidement comment utiliser l'application.

MODÉLISATION UML

Afin d'évaluer correctement nos besoins et de définir la structure de notre projet en C#/.NET Core, nous avons commencé par modéliser le cahier des charges avec des diagrammes UML. Ces diagrammes sont disponibles dans un document rendu en complément de ce présent rapport.

A. Diagramme de cas d'utilisation

Le premier diagramme UML que nous avons modélisé est un diagramme de cas d'utilisation. Il représente, les différentes façons que les utilisateurs vont pouvoir interagir avec le système. Nous avons décidé de détailler les différents cas à l'aide de notes, afin d'éviter les ambiguïtés sur leur utilité.

Nous avons donc spécifié ici les différents cas pour lesquels l'utilisateur pourrait avoir besoin de notre application, ainsi que les cas associés ou bien inclus.

B. Diagramme de classes

Les diagrammes de classes sont utilisés pour représenter la façon dont les classes de notre application sont construites, et à qui elles sont reliées.

Notre diagramme de classe s'organise en trois sous-parties, représentant un modèle d'architecture Model-View-Controller (MVC). Le cadre « Model » contient les classes en rapport avec les données de notre application, comme par exemple les travaux de sauvegarde ou les fichiers contenant les différents langages. Le cadre « View » est associé aux vues de notre application, c'est-à-dire toutes les classes responsables de l'affichage sur la console. Enfin, le cadre « Controller » rassemble toutes les classes qui vont orchestrer les appels des autres classes modèles et vues.

Là encore, nous avons utilisé des notes pour préciser nos choix sur les rôles de nos classes.

C. Diagrammes de séquence

Les diagrammes de séquence vont permettre de détailler l'interaction des objets pour chaque scénario du diagramme de cas d'utilisation. Ils montrent le déroulement logique des différentes méthodes et les échanges dans un ordre chronologique.

Ici nous détaillerons les échanges lorsque l'utilisateur veut créer, supprimer ou exécuter une sauvegarde et lorsqu'il veut choisir une langue. Par soucis de lisibilité, nous avons omis de mettre certains contrôleurs afin d'afficher les fonctions principales du programme.

D. Diagrammes d'activité

Nous avons ensuite réalisé plusieurs diagrammes d'activité afin de détailler les différentes options dans le fonctionnement de notre application. Ces diagrammes nous permettent en effet de constater, un peu à la manière d'un algorithme, à quelles possibilités va être confronté l'utilisateur.

Pour cela, nous avons donc choisi de réaliser un diagramme pour chaque « menu » dans lequel l'utilisateur peut se trouver : le menu principal (dont l'accès se fait par défaut au lancement de l'application) ainsi que le choix de la langue, la création d'un travail de sauvegarde, la suppression de l'un d'entre eux, et l'exécution d'un travail.

Nous avons précisé à chaque fois l'origine des points d'entrée et la destination des points de sortie des diagrammes.

E. Diagrammes de composants

Enfin, nous avons conçu un diagramme de composants, qui permet d'avoir une vue d'ensemble du logiciel et ainsi mieux visualiser l'organisation des composants du système et leurs relations.

Dans ce diagramme, nous avons défini 3 composants constitués d'un ensemble de classes. Chaque composant représente un cas d'utilisation et met en évidence les dépendances entre chaque classe qui le compose, ainsi que les relations entre chaque composant pour indiquer qu'un élément fait appel aux services offerts par les éléments d'implémentation d'un autre composant.

GUIDE UTILISATEUR

Dans cette deuxième partie, nous détaillerons les guides utilisateurs pour aider au bon fonctionnement et à la compréhension de notre application. Ces guides sont proposés en français et en anglais, puisque ce sont les langues disponibles sur EasySave.

En parallèle, ces guides ont été ajoutés au projet GitHub dans un fichier README.

A. Guide francophone

1. Démarrage de l'application

Le démarrage de l'application se fait à l'aide d'un fichier exécutable. En lançant l'application, une console s'ouvre, et on accède au menu principal.

2. Choix de la langue

Lors du lancement de l'application, le choix de la langue est automatiquement demandé. Pour le moment, seuls le français et l'anglais sont disponibles, mais il est à tout moment possible d'en ajouter.

Pour cela, nous avons conçu notre application de manière à ce que les langues soient gérées dans des fichiers précis. Chaque fichier JSON contient les données nécessaires à l'affichage des menus pour un langage. Il est donc tout à fait possible d'en ajouter et de traduire les données, sans avoir besoin de modifier le code source.

3. Création d'un travail de sauvegarde

La première option du menu principal est la création d'un travail de sauvegarde. L'application nous permet d'en créer plusieurs et de stocker ces travaux dans un fichier JSON. Ainsi, ces travaux sont gardés en mémoire au lieu d'être perdus entre chaque démarrage d'EasySave.

Le logiciel demande d'entrer dans l'ordre :

- Un nom de sauvegarde
- Un type de sauvegarde : différentiel ou complet
- L'emplacement source des fichiers à sauvegarder
- L'emplacement de destination souhaité

Une fois la création effectuée, l'utilisateur est automatiquement renvoyé vers le menu principal. Si une information n'est pas correctement entrée, il doit recommencer sa saisie jusqu'à ce qu'elle soit valide.

4. Exécution d'un travail de sauvegarde

La deuxième option du menu principal est l'exécution d'une ou plusieurs sauvegardes. La console affiche la liste des sauvegardes enregistrées ainsi que leurs caractéristiques, et demande si l'utilisateur veut choisir une sauvegarde à exécuter, ou bien exécuter toutes les sauvegardes enregistrées.

Si l'utilisateur choisit de n'en exécuter qu'une, il doit alors entrer le nom de la sauvegarde voulue. Une fois l'exécution effectuée, il est automatiquement renvoyé vers le menu principal. Si une information n'est pas correctement entrée, il doit recommencer sa saisie jusqu'à ce qu'elle soit valide.

5. Suppression d'un travail de sauvegarde

Enfin, la troisième et dernière option du menu principal est la suppression d'une sauvegarde enregistrée. La console affiche la liste des sauvegardes enregistrées ainsi que leurs caractéristiques, et demande à l'utilisateur d'entrer le nom de la sauvegarde qu'il veut supprimer.

Une fois la suppression effectuée, il est automatiquement renvoyé vers le menu principal. Si une information n'est pas correctement entrée, il doit recommencer sa saisie jusqu'à ce qu'elle soit valide.

6. Journaux d'activité

Les journaux d'activités sont créés pour permettre à l'utilisateur de vérifier à tout moment le bon déroulement de ses sauvegardes. Il existe deux types de journaux : le premier répertorie toutes les sauvegardes correctement exécutées durant la journée, et le second affiche en temps réel l'avancement des sauvegardes.

Une fois créés, ces fichiers sont disponibles dans le dossier « bin/Debug/netcoreapp3.1/ ».

B. Guide anglophone

1. Starting the application

The application is started using an executable file. When launching the application, a console opens, and the main menu is accessed.

2. Choosing the language

When the application is launched, the choice of language is automatically requested. For the moment, only French and English are available, but it's possible to add more languages at any time.

In order to do this, we have designed our application so that the languages are managed in specific files. Each JSON file contains the data necessary to display the menus for a language. It is therefore entirely possible to add and translate the data, without having to modify the source code.

3. Creating a job

The first option in the main menu is the creation of a backup job. The application allows us to create several jobs and store them in a JSON file. This way, these jobs are kept in memory instead of being lost between each start of EasySave.

The software asks to enter in order :

- A backup name
- A backup type: differential or full
- The source location of the files to backup
- The desired destination location

Once the creation is complete, the user is automatically returned to the main menu. If a piece of information is not correctly entered, the user must repeat the entry until it is valid.

4. Running a job

The second option in the main menu is to run one or more backups. The console displays a list of saved backups and their characteristics, and asks if the user wants to choose one backup to run, or run all saved backups.

If the user chooses to run only one backup, they must enter the name of the backup they want to run. Once executed, the user is automatically returned to the main menu. If any information is entered incorrectly, the user must re-enter it until it is valid.

5. Deleting a job

Finally, the third and last option in the main menu is to delete a saved backup. The console displays a list of saved backups and their characteristics, and asks the user to enter the name of the backup they want to delete.

Once deleted, the user is automatically returned to the main menu. If a piece of information is not correctly entered, the user must repeat the entry until it is valid.

6. Log files

Activity logs are created to allow the user to check at any time that their backups are running correctly. There are two types of logs: the first lists all the backups that were successfully executed during the day, and the second displays the progress of the backups in real time.

Once created, these files are available in the « bin/Debug/netcoreapp3.1/ » folder.

CONCLUSION

Pour ce premier livrable, nous avons conçu une application console prête à être utilisée par des utilisateurs français et anglais. Elle permet de créer des travaux de sauvegarde et de les exécuter afin de copier des fichiers d'un dossier source vers un dossier de destination.

Nous avons réussi à modéliser notre projet à l'aide de plusieurs diagrammes UML. Nous avons ainsi évalué toutes les possibilités et spécificités d'exécutions de notre application, et nous avons mis en applications des design patterns pour que notre code soit correctement structuré. Le guide utilisateur que nous avons mis en ligne sur notre projet GitHub permettra à n'importe qui de pouvoir utiliser facilement l'application, et éventuellement de reprendre le projet s'il venait à être modifié.

Nous avons désormais une application console fonctionnelle en C#. Dans la suite du projet, nous pourrons petit à petit faire évoluer ce logiciel en une application gérée par une interface.

GLOSSAIRE

N°	Nom ou acronyme	Définition
1	Job	Le mot « *job* » fait référence à la sauvegarde avec tous ses attributs (Nom, Type, Chemin de la source/destination …).
2	Model	Le modèle est l'élément central de l'application, il permet de gérer les données, la logique et les règles de l'application.
3	View	La vue est le moyen d'afficher des objets dans l'application comme le texte dans une fenêtre, des boutons, des icônes...
4	Controller	Le contrôleur est l'articulation de l'application, c'est lui qui met à jour les vues et les modèles.

BIBLIOGRAPHIE

N°	Lien de la source	Contenu
1	https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/;WWWSESSIONID=32D134237A63249355388A0504711C9D.www1	VISUAL PARADIGM. ***What is Component Diagram ?*** [En ligne]
2	https://refactoring.guru/fr/design-patterns/singleton	REFACTORING GURU. ***Singleton.*** [En ligne]
3	https://stackoverflow.com/questions/9497968/how-to-show-a-singleton-relationship-in-a-class-diagram	STACK OVERFLOW. ***How to show a Singleton relationship in a class diagram.*** 29 février 2012. [En ligne]