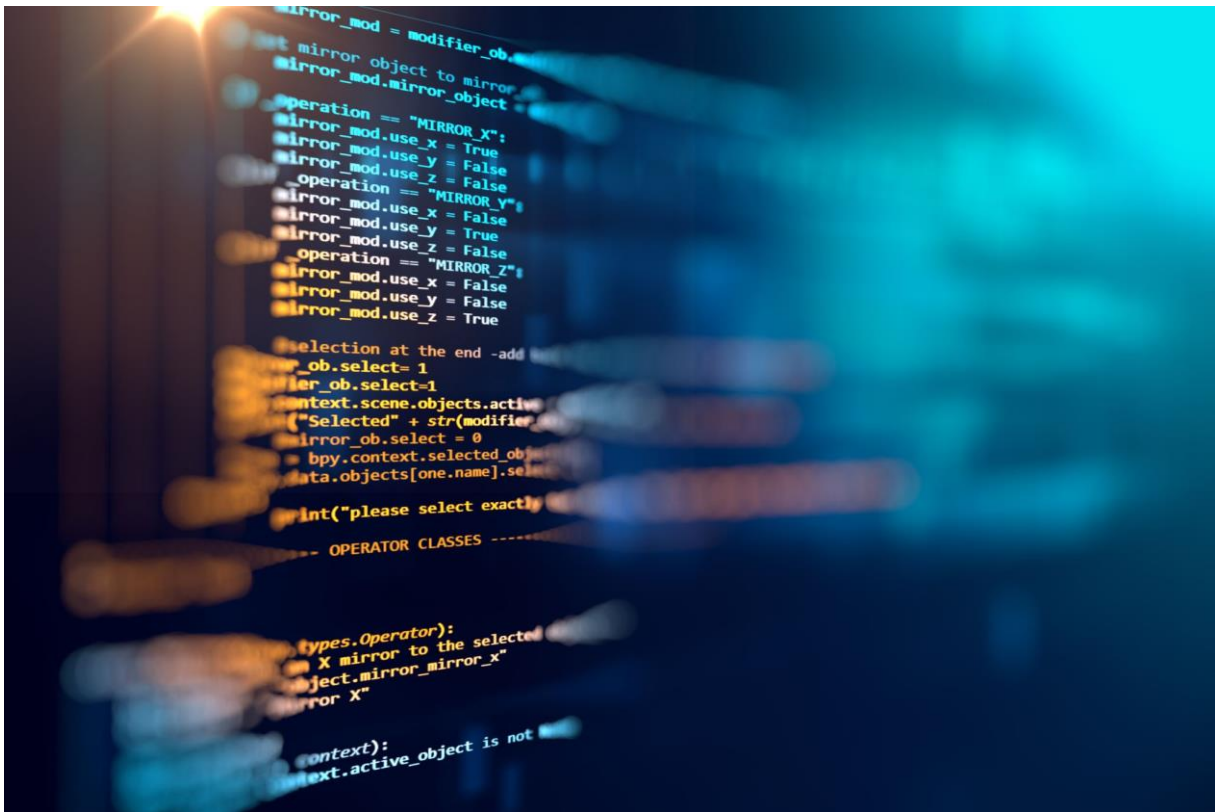


Programmation système

Livrable 1 : Diagramme UML du logiciel EasySave



Membre du groupe :

Jawad

Perla

Abel

Florent

Table des matières

Introduction	2
Description du livrable 0 et 1 : EasySave version 1.0.....	3
Présentation des diagrammes UML.....	5
Diagramme de cas d'utilisation	5
Diagramme d'activité	6
Diagramme de class	9
Diagramme de séquence	10

Introduction

Ce livrable présente la phase d'analyse pour le développement de la première version du logiciel EasySave. Notre est tenue d'installer un environnement de travail respectant les contraintes imposées par ProSoft. La gestion de GIT (versioning), les diagrammes UML et la qualité du code seront des aspects particulièrement surveillés tout au long du projet.

La première version du logiciel, EasySave 1.0, est une application Console développée à l'aide de .Net Core. Elle permettra à l'utilisateur de créer jusqu'à 5 travaux de sauvegarde définis par leur appellation, répertoire source, répertoire cible et type (complet ou différentiel). Le logiciel devra être utilisable par des utilisateurs anglophones et francophones.

Les répertoires de sauvegarde peuvent être sur des disques locaux, externes ou sur des lecteurs réseaux et tous les éléments du répertoire source seront inclus dans la sauvegarde. Le logiciel écrira en temps réel dans un fichier log journalier les actions des travaux de sauvegarde, ainsi que dans un fichier unique, l'état d'avancement des travaux de sauvegarde. Les fichiers seront au format JSON et il sera nécessaire de les formater pour une lecture rapide via Notepad.

Enfin, il est important de noter que si le logiciel donne satisfaction, la direction demandera de développer une version 2.0 utilisant une interface graphique WPF basée sur l'architecture MVVM.

Nous espérons que ce livrable vous fournira une bonne compréhension de l'environnement de travail et de la première version du logiciel EasySave.

Description du livrable 0 et 1 : EasySave version 1.0

Version 1.0

Description du livrable 0 : Environnement de travail

Votre équipe doit installer un environnement de travail respectant les contraintes imposées par ProSoft.

Le bon usage de l'environnement de travail et des contraintes imposées par la direction seront évalués tout au long du projet.

Une vigilance particulière sera portée sur :

- La gestion de GIT (versioning)
- Les diagrammes UML à rendre 24 heures avant chaque livrable (Jalon)
- La qualité du code (absence de redondance dans les lignes de code)

Description du livrable 1 : EasySave version 1.0

Le cahier des charges de la première version du logiciel est le suivant :

- Le logiciel est une application Console utilisant .Net Core.
- Le logiciel doit permettre de créer jusqu'à 5 travaux de sauvegarde
- Un travail de sauvegarde est défini par
 - Une appellation
 - 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.
- Les répertoires (sources et cibles) pourront être sur :
 - Des disques locaux
 - Des disques Externes
 - Des Lecteurs réseaux
- Tous les éléments du répertoire source sont concernés par la sauvegarde
- Fichier Log 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
 - Appellation du travail de sauvegarde
 - Adresse complète du fichier Source (format UNC)
 - Adresse complète du fichier de destination (format UNC)
 - Taille du fichier

- Temps de transfert du fichier en ms (négatif si erreur)
- Exemple de contenu : Sample_log.pdf [pdf]
- 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
 - Etat du travail de Sauvegarde (ex : Actif, 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
- Exemple de contenu : Sample_state.pdf [pdf]
- Les emplacements des deux fichiers (log journalier et état) devront être étudiés pour fonctionner sur les serveurs des clients. De ce fait, les emplacements du type « c:\temp\ » sont à proscrire.
- Les fichiers (log journalier et état) 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)

Présentation des diagrammes UML

Le **diagramme UML (Unified Modeling Language)** est un langage de modélisation graphique utilisé pour la conception et la documentation des systèmes informatiques. Il permet de représenter les différents aspects d'un système, tels que les objets, les classes, les relations entre objets, les processus, les flux de données, etc.

Les différents types de diagrammes UML incluent :

Diagramme de cas d'utilisation : représente les actions que les utilisateurs peuvent effectuer avec le système.

Diagramme de classes : représente les classes et les relations entre elles dans le système.

Diagramme de séquence : représente les interactions entre les objets au cours du temps.

Diagramme de collaboration : représente les relations entre les objets et comment ils coopèrent pour réaliser une tâche.

Diagramme de statut : représente les différents états dans lesquels peut se trouver un objet et les transitions entre ces états.

Diagramme de composants : représente les différents composants logiciels qui constituent le système et comment ils sont connectés.

Diagramme de déploiement : représente les éléments matériels et logiciels qui forment l'environnement de déploiement du système.

Dans le cadre de ce projet nous avons :

Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation est un type de diagramme UML qui représente les actions que les utilisateurs peuvent effectuer avec le système. Il permet de visualiser les fonctionnalités du système et comment les utilisateurs interagissent avec celui-ci.

Dans un diagramme de cas d'utilisation, les cas d'utilisation sont représentés sous forme de cercles ou de boîtes, qui décrivent les actions réalisées par l'utilisateur. Les acteurs, qui représentent les utilisateurs du système, sont représentés sous forme de stickman. Les flèches reliant les cas d'utilisation aux acteurs indiquent les interactions entre eux.

L'application d'un diagramme de cas d'utilisation est la suivante :

Clarification des exigences du système : en identifiant les cas d'utilisation, il est possible de mieux comprendre les fonctionnalités nécessaires pour répondre aux besoins des utilisateurs.

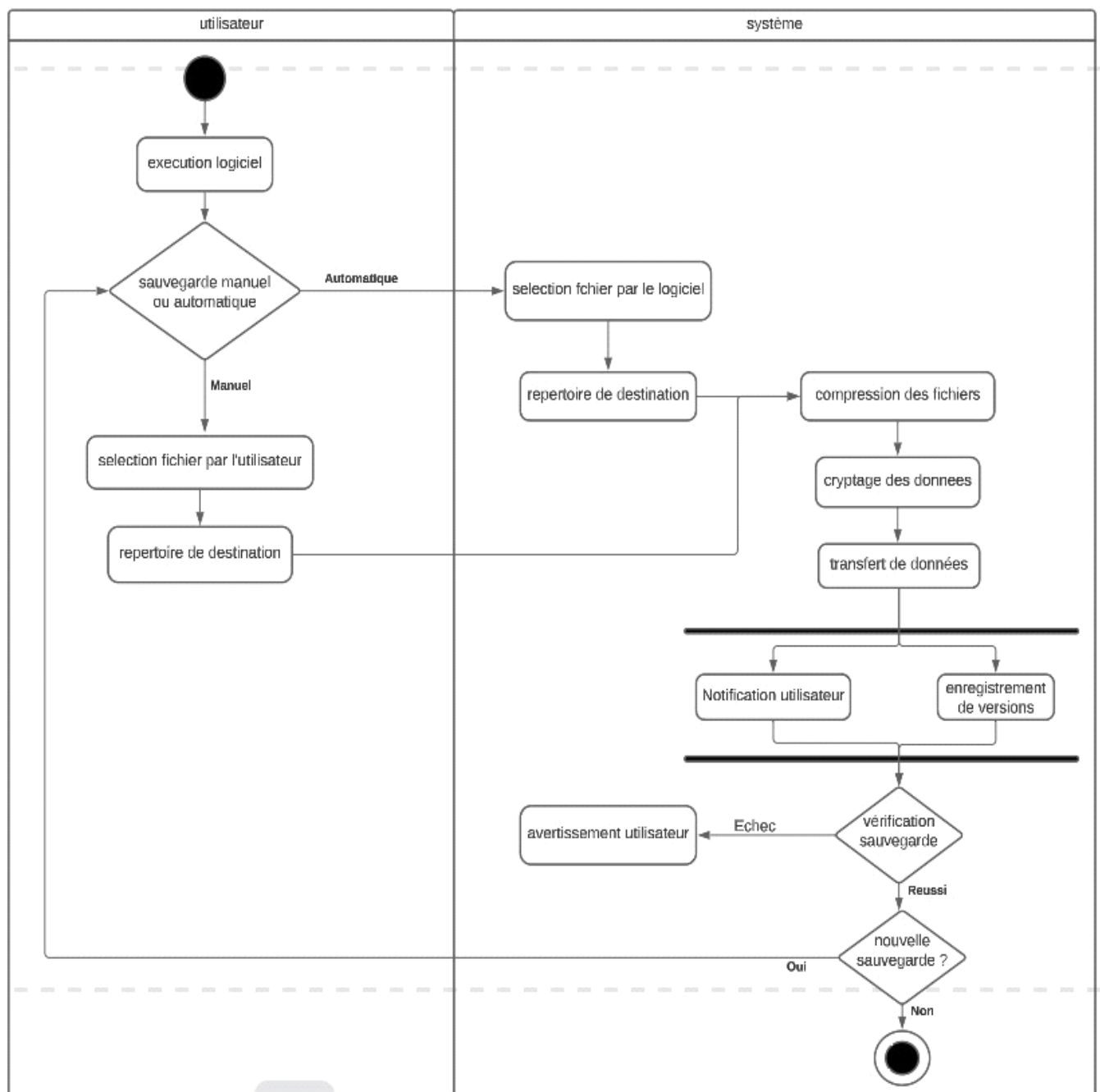
Communication avec les parties prenantes : les diagrammes de cas d'utilisation peuvent être utilisés pour communiquer avec les différentes parties prenantes, telles que les développeurs, les utilisateurs, les responsables de la qualité, etc.

Maintenance et évolution : les diagrammes de cas d'utilisation peuvent être utilisés pour documenter les modifications apportées au système au fil du temps, facilitant ainsi la maintenance et l'évolution.



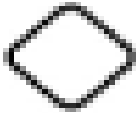
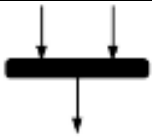

```
graph LR
    ActorName[ActorName]
    subgraph Logiciel_de_sauvegarde [Logiciel de sauvegarde]
        UC1(Choisir un dossier source et un dossier destination)
        UC2(Indiquer les informations de sauvegarde)
        UC3(Créer des sauvegarde)
        UC4(sauvegarder en temps réel l'état de progression des tâches de sauvegardes)
        UC5(indiquer les informations sur l'état de progression)
        UC6(indiquer l'extension du fichier de journal)
        UC7(exécuter le travail de sauvegarder)
        UC8(Créer un fichier journal)
        UC9(Lancer une sauvegarde)
        UC10(voir l'état d'avancement)
        UC11(choisir la langue)
        UC12(choisir français)
        UC13(choisir anglais)
    end
    ActorName --- UC1
    ActorName --- UC2
    ActorName --- UC3
    ActorName --- UC4
    ActorName --- UC5
    ActorName --- UC6
    ActorName --- UC7
    ActorName --- UC8
    ActorName --- UC9
    ActorName --- UC11
    UC3 -.->|<<include>>| UC1
    UC3 -.->|<<include>>| UC2
    UC3 -.->|<<include>>| UC4
    UC3 -.->|<<include>>| UC5
    UC3 -.->|<<include>>| UC6
    UC7 -.->|<<include>>| UC8
    UC9 -.->|<<include>>| UC10
    UC11 -.->|<<extend>>| UC12
    UC11 -.->|<<extend>>| UC13
    UC8 -.->|<<include>>| UC13
```

Dans le contexte d'une application, un diagramme d'activité peut aider à visualiser les différentes étapes de l'utilisation de l'application, les interactions entre les différentes parties du système et les processus de traitement qui se produisent en arrière-plan. Cela peut être utile pour la conception, le développement, le test et la maintenance de l'application.

Les éléments couramment utilisés dans un diagramme d'activité comprennent des activités, des décisions, des transitions, des jonctions et des zones de départ et d'arrivée. Les activités représentent les tâches ou les opérations effectuées au sein du système, les décisions représentent les points où le cheminement du processus peut diverger en fonction d'une condition donnée, les transitions



Elément constitutifs du diagramme :

Symboles	Fonctions
	Représente le début d'un processus
	Indique les activités qui composent un processus modélisé.
	Représente une décision et possède toujours au moins deux embranchements avec le texte de la condition pour permettre aux utilisateurs de voir les options.
	Associe deux activités simultanées et les réintroduit dans un flux où n'a lieu qu'une seule activité à la fois. Représenté par une ligne verticale ou horizontale épaisse.
	Marque l'état final d'une activité et représente l'achèvement de tous les flux d'un procédé.

Pour ce diagramme, après exécution du logiciel par l'utilisateur, on a :

1. **Déclenchement de la sauvegarde** : l'utilisateur lance la sauvegarde manuellement ou la sauvegarde est automatisée selon un horaire programmé.
2. **Sélection des fichiers à sauvegarder** : l'utilisateur sélectionne les fichiers à sauvegarder ou le logiciel sélectionne automatiquement tous les fichiers en fonction des paramètres définis par l'utilisateur.
3. **Compression des fichiers** : les fichiers sélectionnés sont compressés pour minimiser la quantité de données à transférer.
4. **Cryptage des données** : les données compressées sont cryptées pour protéger la confidentialité des données cryptées.
5. **Transfert des données** : les données cryptées sont transférées vers le stockage de sauvegarde, soit en local, soit en ligne.
6. **Enregistrement de la version** : une nouvelle version de la sauvegarde est enregistrée avec une date et un numéro de version.
7. **Gestion des versions** : le logiciel de gestion des versions permet de conserver plusieurs versions de la sauvegarde, en permettant à l'utilisateur de restaurer une version antérieure des données tout en étant notifié.

8. **Vérification de la sauvegarde** : une vérification est effectuée pour s'assurer que les données ont été correctement enregistrées et que la sauvegarde est opérationnelle.

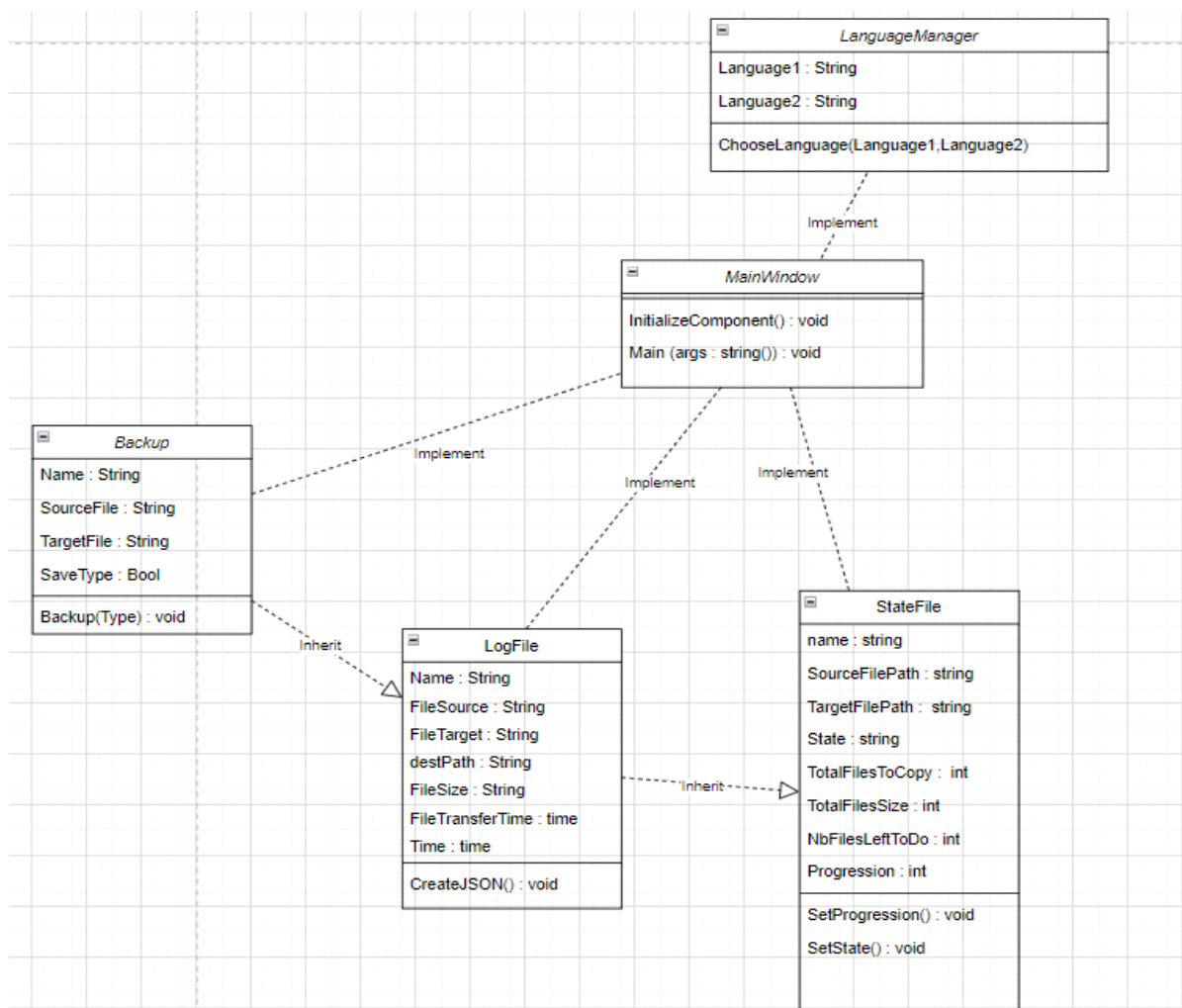
9. **Avertissement en cas d'échec** : en cas d'échec de la sauvegarde, un avertissement est envoyé à l'utilisateur pour informer de la nécessité de vérifier les paramètres et de relancer la sauvegarde.

10. **Fin** : la sauvegarde est terminée et les données sont prêtes à être restaurées en cas de besoin.

Diagramme de classes

Un diagramme de classes UML est utilisé pour représenter les classes, les objets et les interactions entre ces derniers dans un système logiciel. Chaque classe est représentée par un rectangle contenant le nom de la classe, les attributs et les méthodes. Les relations entre les classes sont représentées par des flèches reliant les classes concernées.

Il existe plusieurs types de relations dans les diagrammes de classes UML, tels que l'association, l'agrégation, la composition et l'héritage. Chacun de ces types de relations est représenté par un type spécifique de flèche et aide à décrire la manière dont les classes sont liées les unes aux autres.



LanguageManager est une classe pour changer la langue, contenant deux attributs langue et une méthode pour changer l'affichage soit en français soit en anglais.

MainWINDOW c'est une classe reprenant notre programme dans sa globalité

Backup est la classe prenant en attribut les paramètres de notre futur travail de sauvegarde

StateFile c'est la classe qui représente l'état du travail de sauvegarde

LogFile cest la classe des paramètres qui représente le dossier source

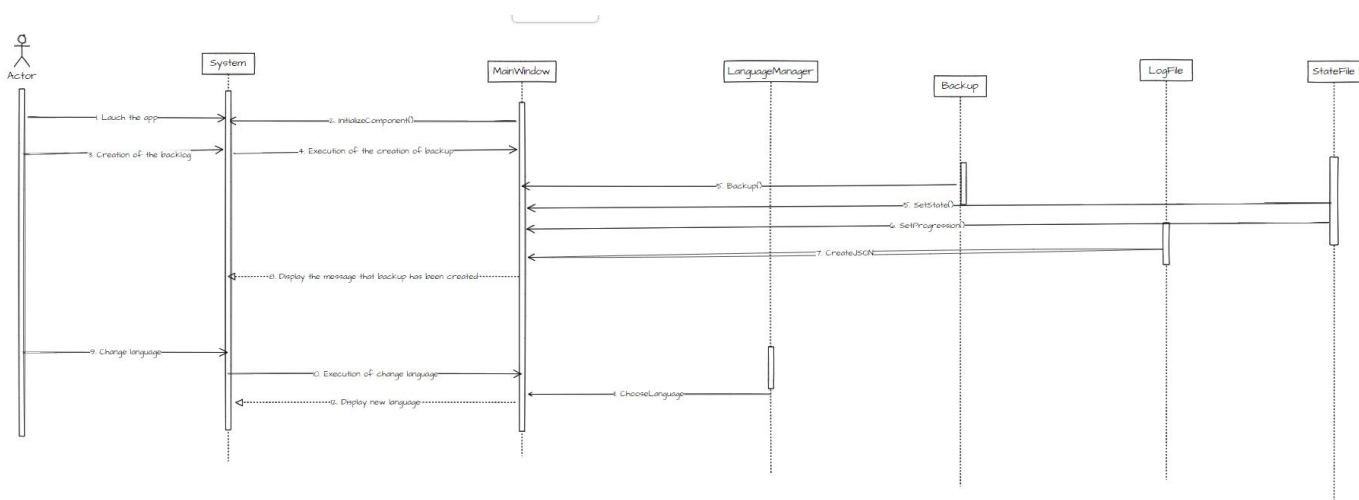
Diagramme de séquence

Les diagrammes de séquence UML sont souvent utilisés pour décrire les scénarios d'interaction entre les objets d'une application, en montrant les messages envoyés entre les objets, le moment où ils sont envoyés et la séquence dans laquelle ils se produisent. Cela peut aider à visualiser et à comprendre les interactions complexes entre les objets dans une application.

Dans l'application, un diagramme de séquence peut être utilisé pour documenter les interactions entre les différents composants d'une application, pour aider à déterminer les points de blocage et les problèmes de performance potentiels, pour clarifier les interactions entre les utilisateurs et l'application, et pour aider à déterminer les scénarios de test.

Les éléments couramment utilisés dans un diagramme de séquence UML comprennent des objets, des flèches représentant les messages envoyés entre les objets, et des numéros de séquence associés aux messages pour indiquer l'ordre dans lequel ils se produisent. Les objets peuvent être représentés sous la forme de boîtes contenant le nom de l'objet et, éventuellement, d'autres informations telles que les attributs ou les méthodes associées.

Le fonctionnement de notre application de manière séquentiel se déroulera comme sur le diagramme indiqué en dessous :



L'utilisateur démarrera l'application qui à ce moment-là chargera les différents paramètres nécessaires à son bon fonctionnement, une fois que ça sera fait elle présentera plusieurs options à l'utilisateur comme la possibilité de commencer un travail de sauvegarde ou changer la langue de l'affichage.

Quand l'utilisateur demande à l'application un travail de sauvegarde plusieurs fonctions s'enchainent, la fonction backup fait appel aux méthodes « SetState » et « StateProgression » qui permettront à l'utilisateur de renseigner les différents paramètres qu'il souhaite pour son travail de sauvegarde.

Lorsque ces informations sont entrées et validées par le programme, la méthode « CreateJSON » est appelé afin de créer un fichier correspondant aux performances ou caractéristiques du travail créé. Si le fichier est bien créé alors l'application affiche un message de confirmation à l'utilisateur.

Pour changer de langue l'utilisateur doit appuyer sur un bouton se trouvant sur l'IHM, lorsque ce bouton est actionné il appelle la méthode « ChooseLanguage » qui vérifie la langue actuelle de l'affichage et charge une version de l'application avec une nouvelle langue

Conclusion

En conclusion ce livrable nous a permis dans un premier temps de mieux comprendre le projet puis de ressortir des compétences en analyse de système d'information et enfin d'apprendre la conception UML ainsi que le modèle MVVC. L'analyse étant fait la prochaine étape sera de passé à l'élaboration de la 1^{ère} version de l'application

Bibliographie

[Livrable 0 et 1 \(cesi.fr\)](https://cesi.fr/)

[Les types de diagrammes UML | Blog Lucidchart](#)

[Qu'est-ce que le langage UML | Lucidchart](#)