Livrable I

PROJET Programmation système

Tuteur

HAMOUR Nora

Réalisé par le Groupe VI

RESSEGAIRE DAMIEN
RAVE ANTOINE
MATHIEU VIRGILE
ROUX KÉVIN

Table des matières

Contexte

Notre équipe vient d'intégrer l'éditeur de logiciels ProSoft. Sous la responsabilité du DSI, nous avons la responsabilité de gérer le projet “EasySave” qui consiste à développer un logiciel de sauvegarde. Lors de ce projet, notre équipe doit assurer le développement, la gestion des versions majeures et mineures, mais aussi les documentations.

Présentation du groupe

Notre groupe est composé de 4 membres : RESSEGAIRE DAMIEN, RAVE ANTOINE, MATHIEU VIRGILE, ROUX KÉVIN.

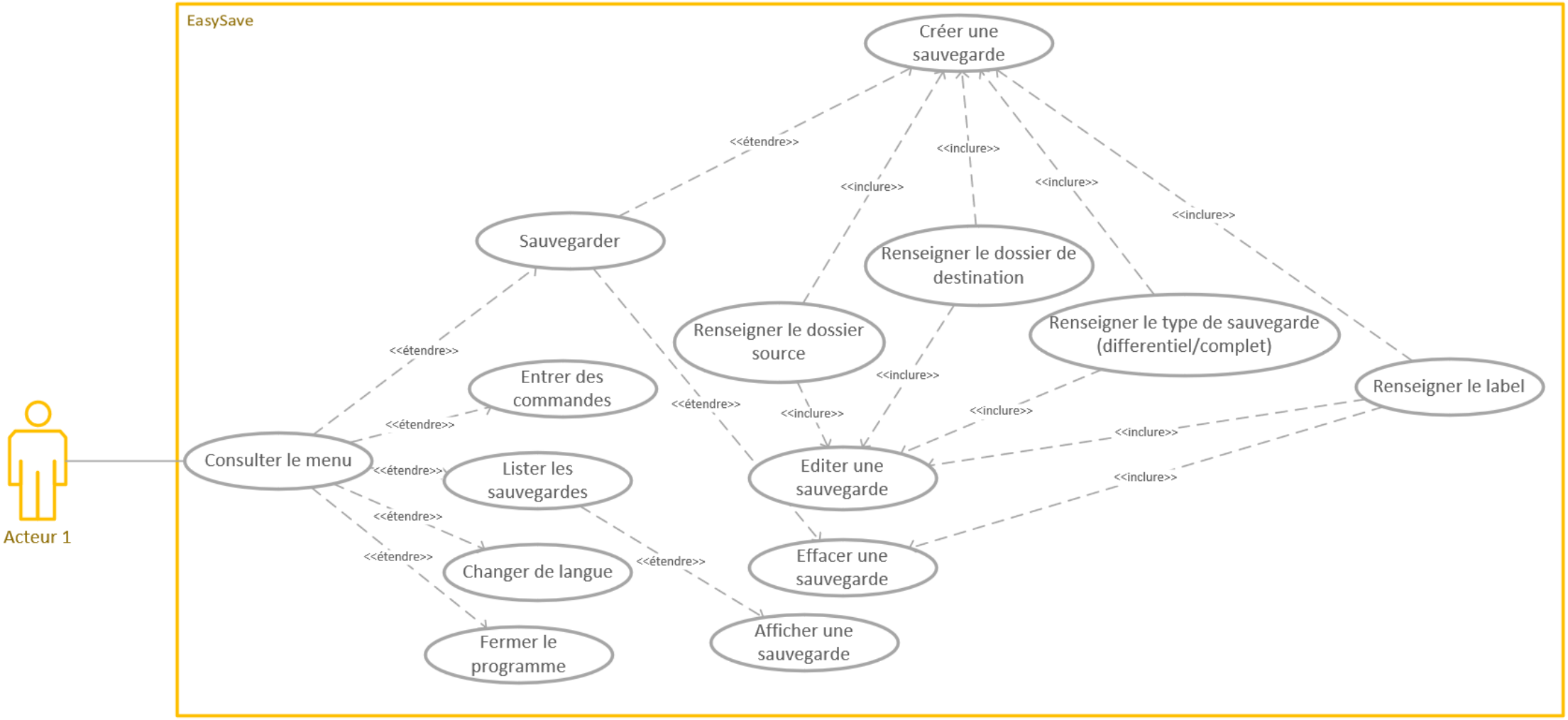
Introduction

Dans ce livrable nous vous présentons notre modélisation pour le projet EasySave. Nous avons effectué des diagramme UML de type diagramme use case, diagramme de séquence, diagramme d’activité et diagramme de classe. Puis nous avons codé nôtre application console. Enfin nous avons rédigé une documentation utilisateur.

I) Diagramme UML

A) Diagramme Use Case

Tout d'abord nous avons décidé de créer un diagramme de cas d'utilisation afin d'avoir une vision claire et globale du projet. Pour cela nous avons décidé de créer un diagramme de cas d'utilisation général donc de très haut niveau.

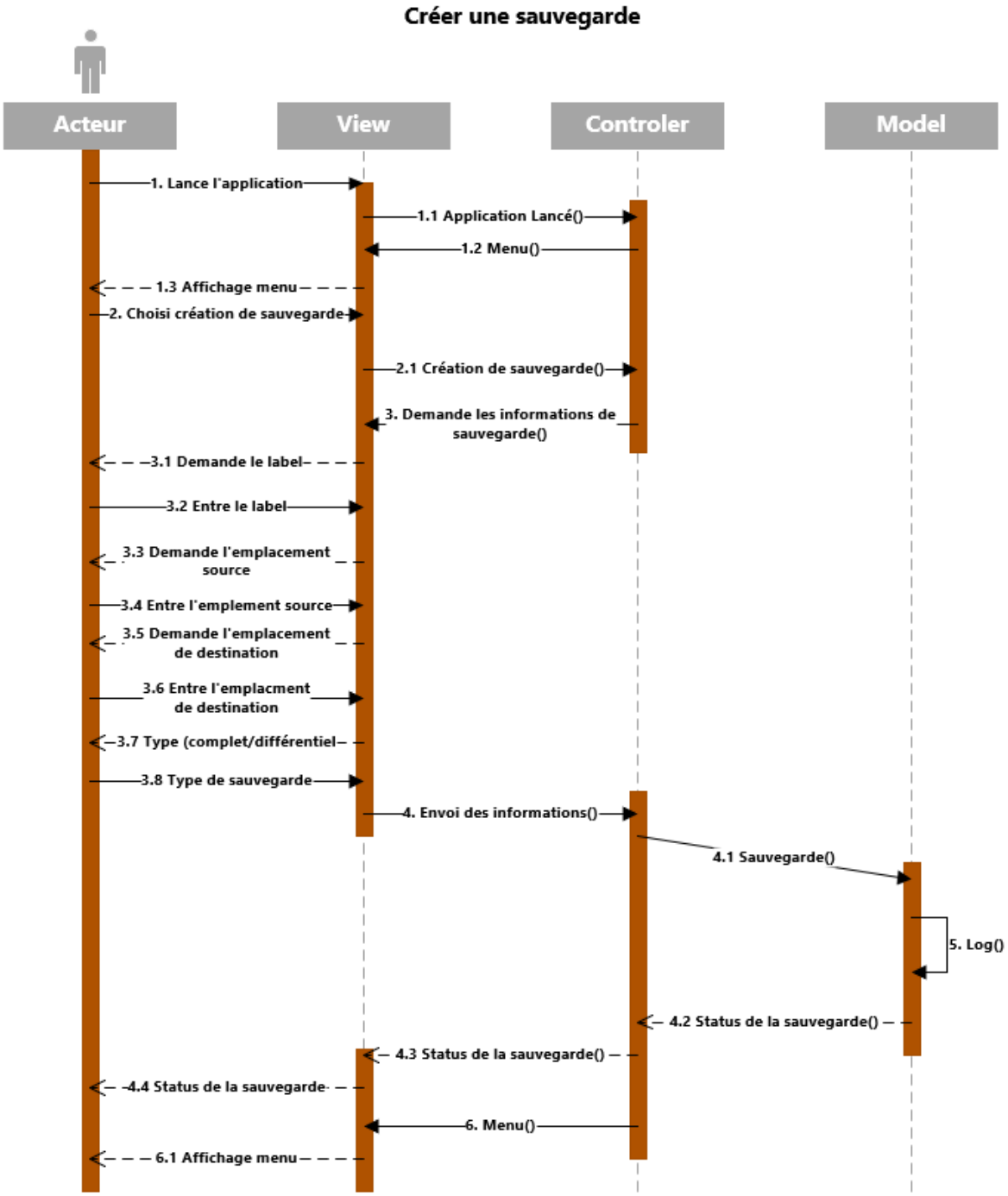


Notre acteur a donc 5 cas d'utilisation qui sont : sauvegarder, enter des commandes, lister les sauvegardes, changer de langue, fermer le programme

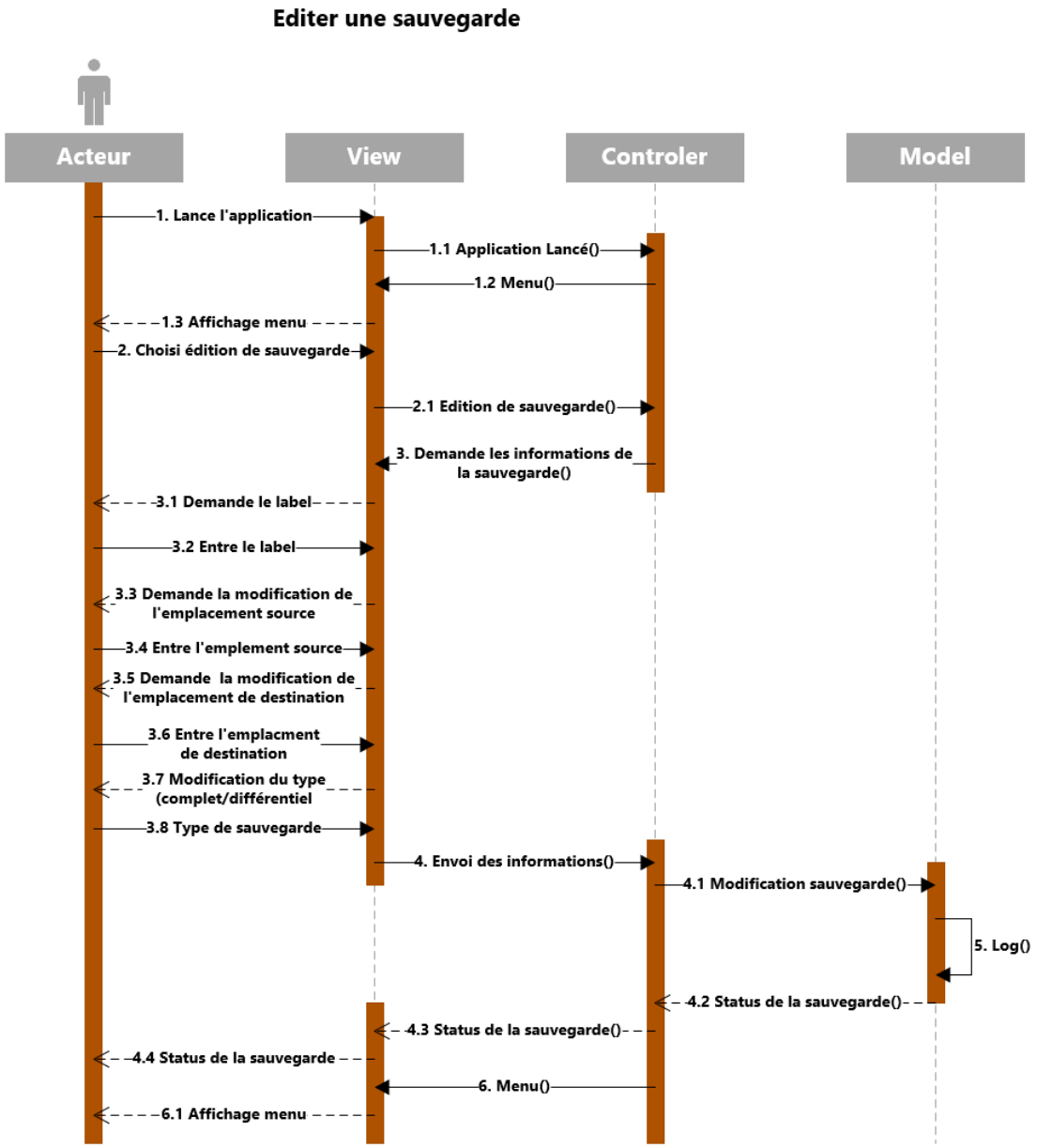
B) Diagramme séquences

Nous avons ensuite créé les diagrammes séquences en fonctions des différentes fonctionnalité.

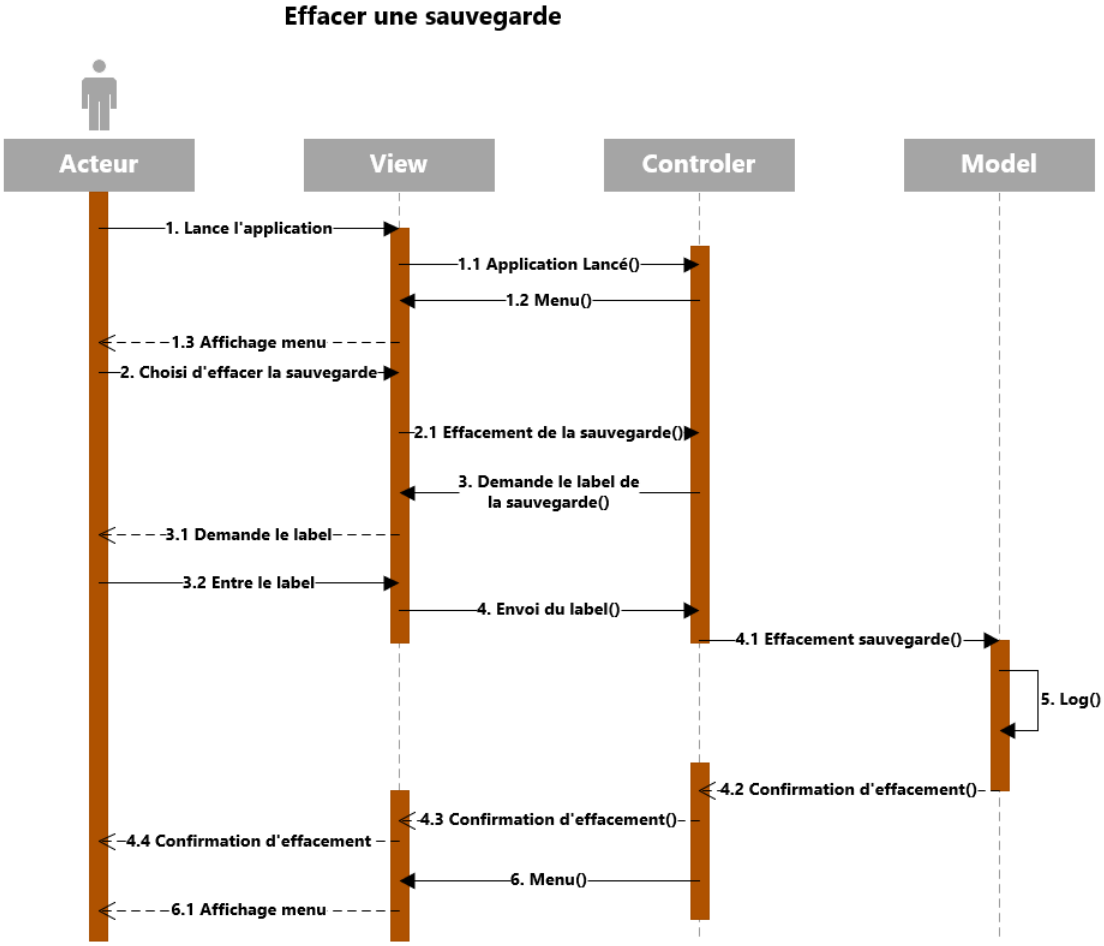
1) Créer une sauvegarde



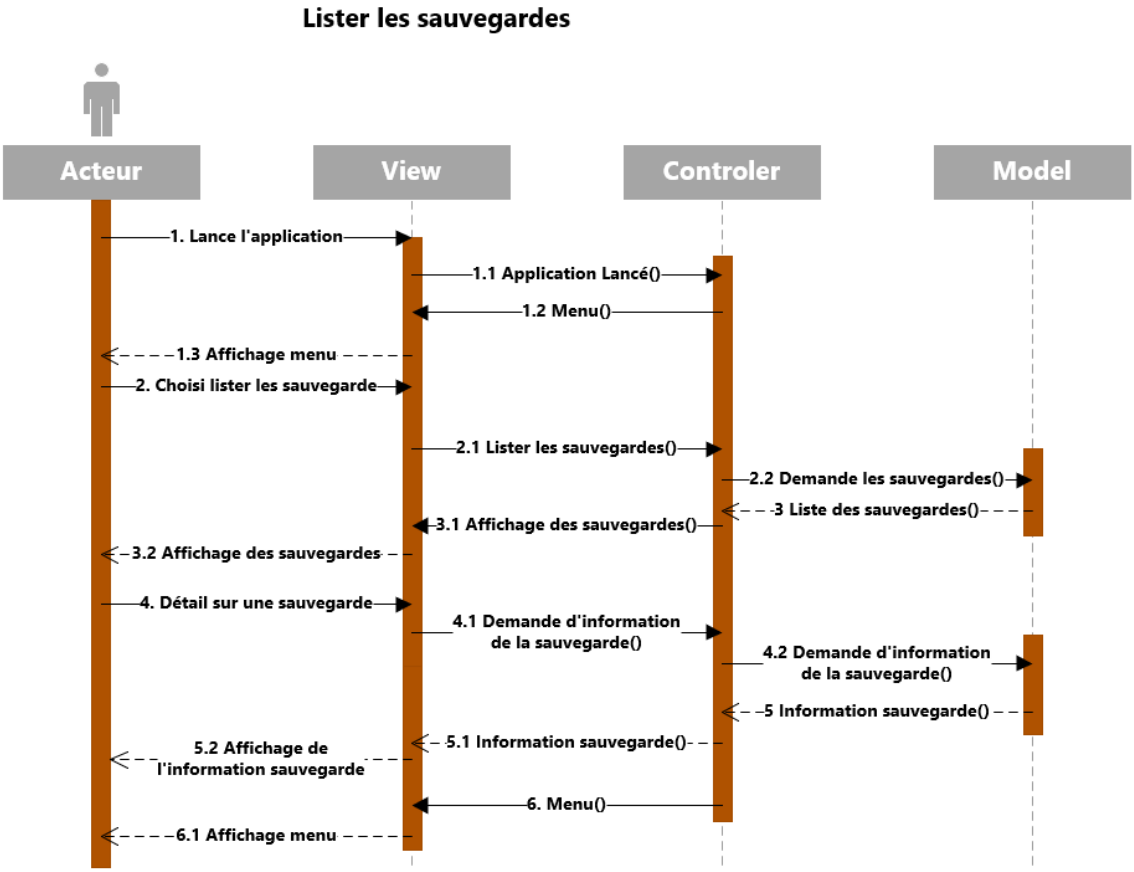
2) Editer une sauvegarde



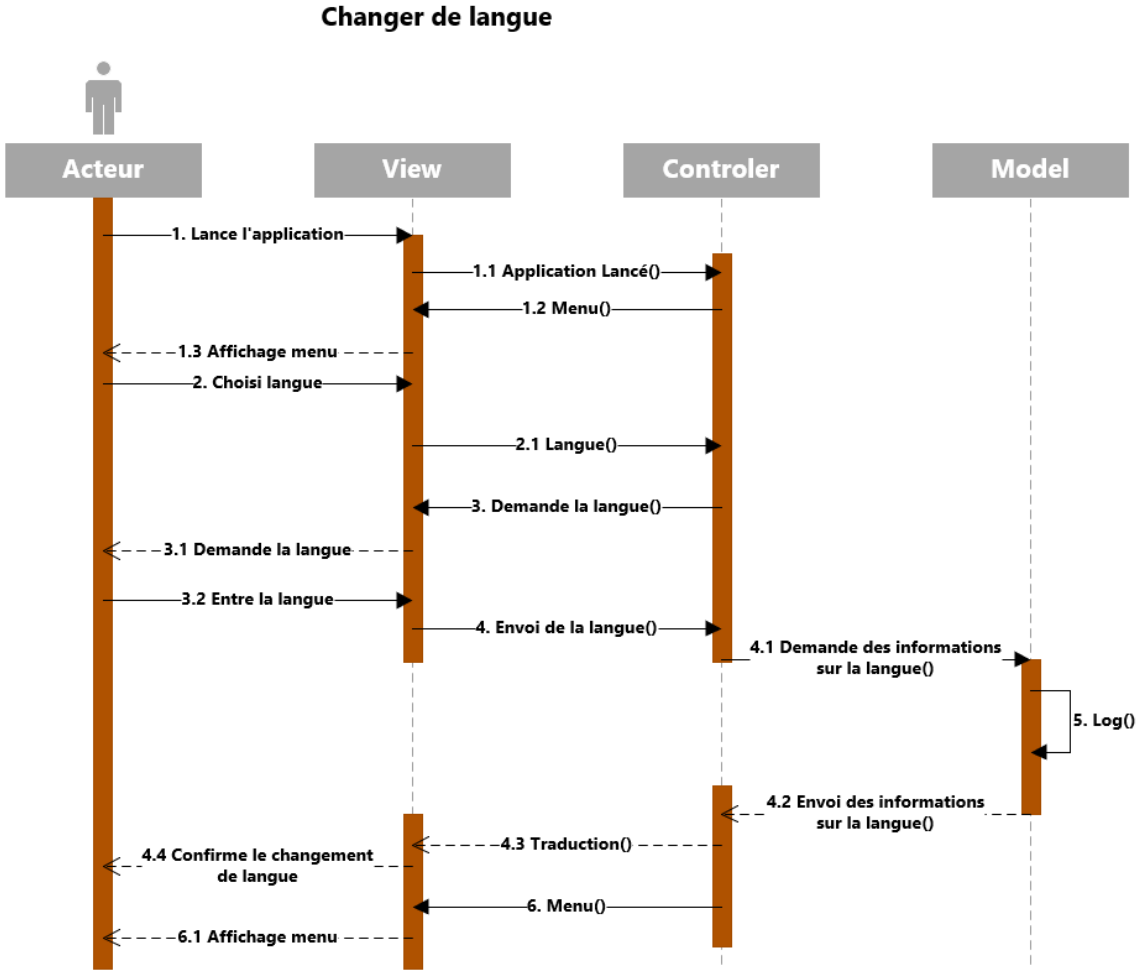
3) Effacer une Sauvegarde



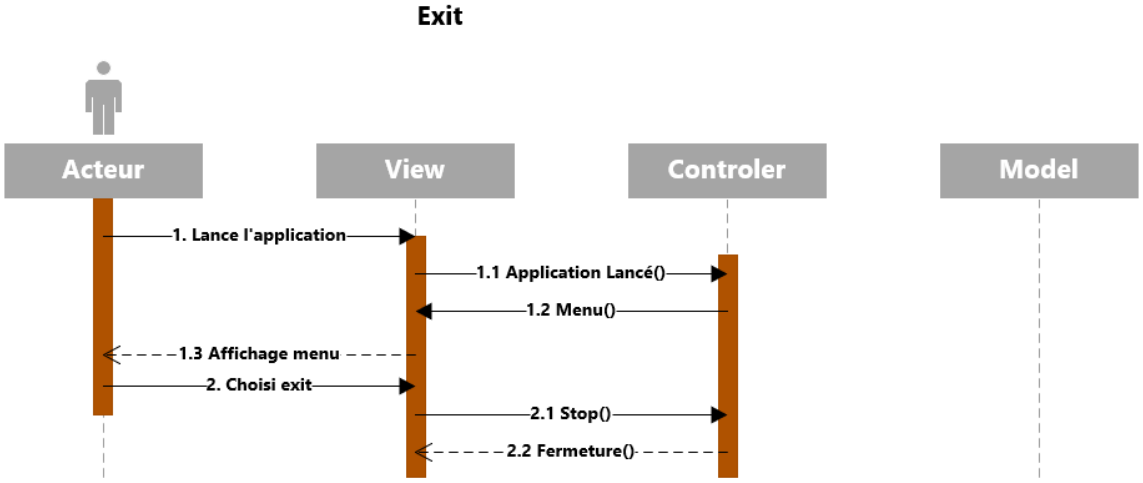
4) Lister les sauvegardes



5) Changer de langue



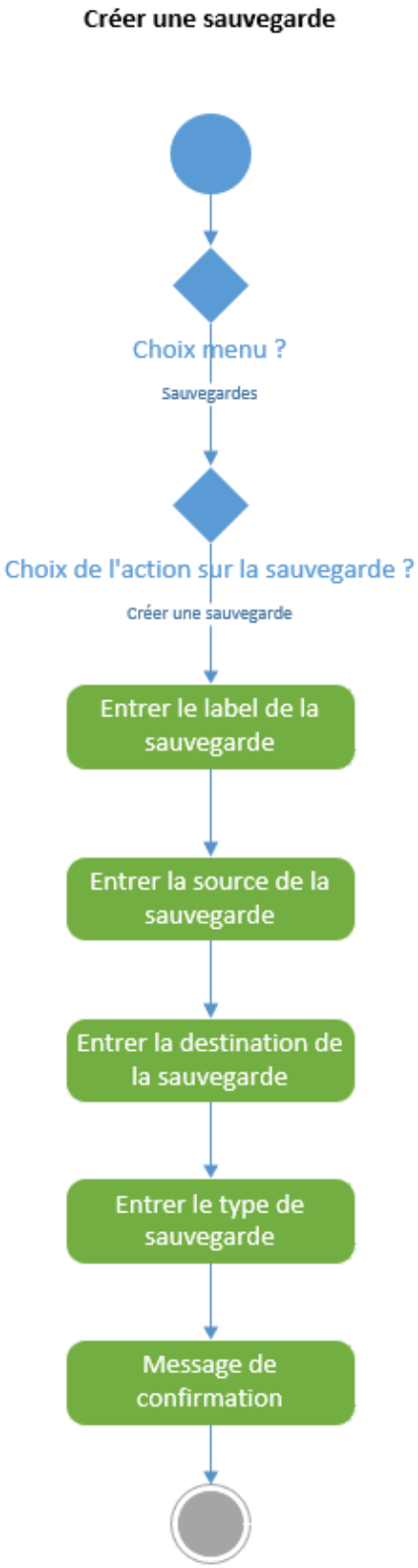
6) Quitter



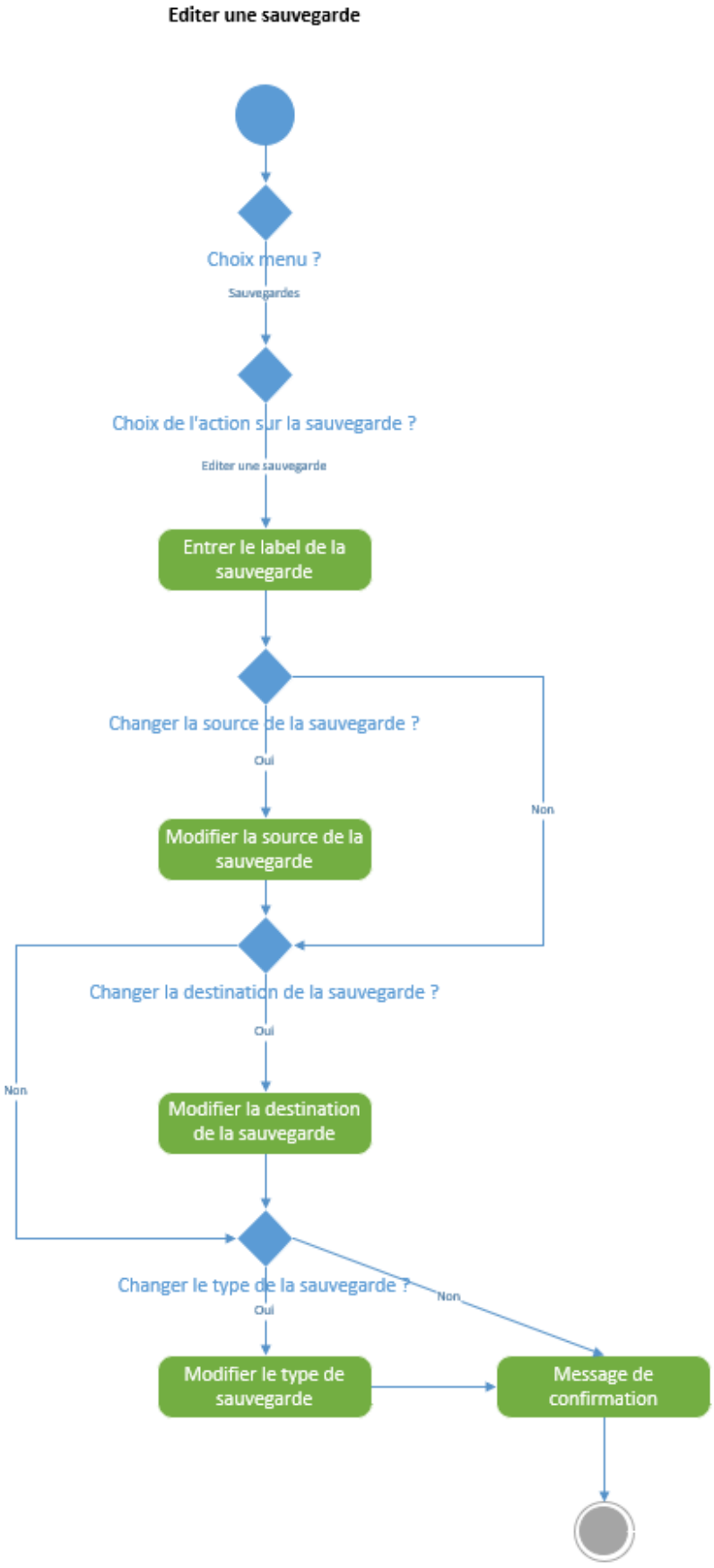
C) Diagramme d'activité

Nous nous sommes ensuite penchés sur l'enchaînement des actions dans notre interface. Pour cela, nous avons utilisé les diagrammes d'activité.

1) Créer une sauvegarde

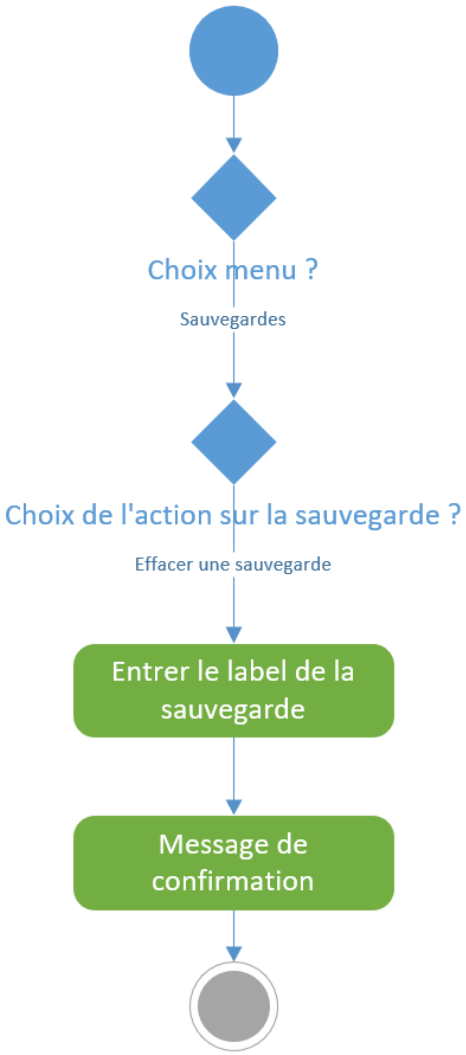


2) Editer une sauvegarde



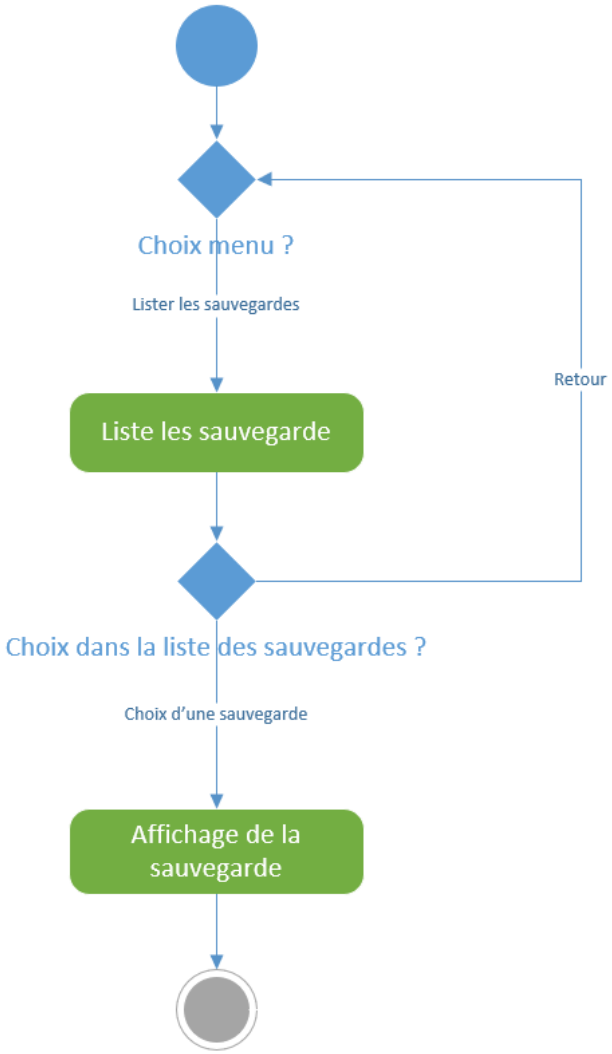
3) Effacer une sauvegarde

Effacer une sauvegarde

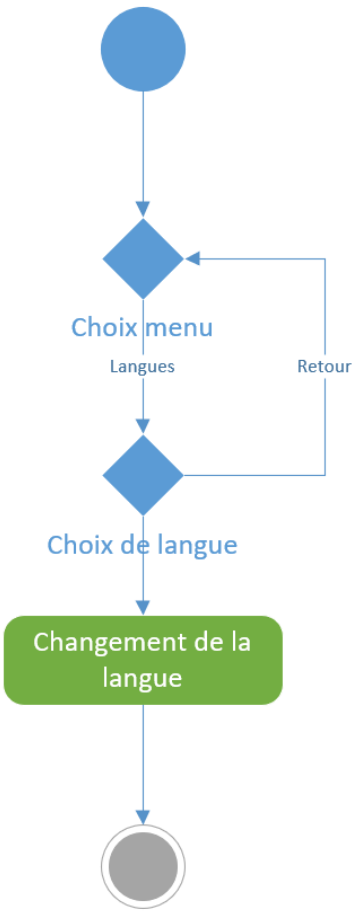


4) Lister les sauvegardes

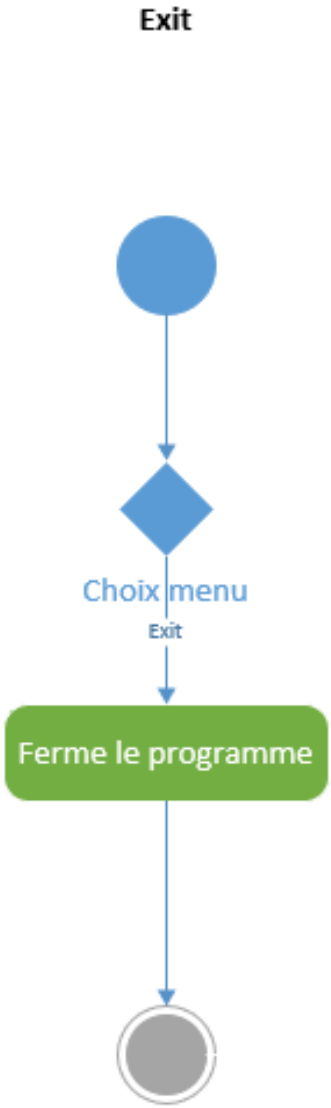
Lister les sauvegardes



5) Changer de langue



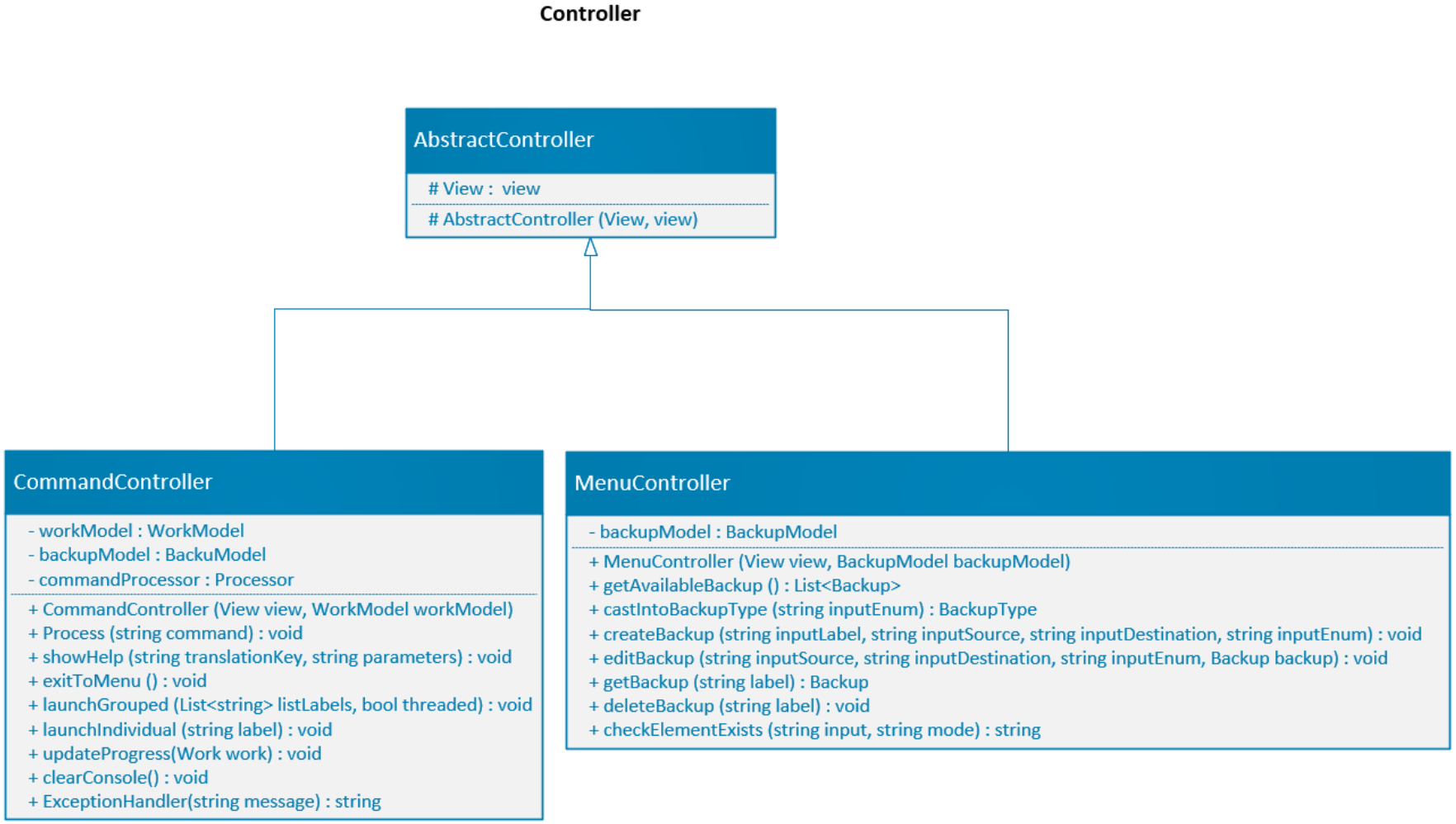
6) Quitter



D) Diagramme de classe

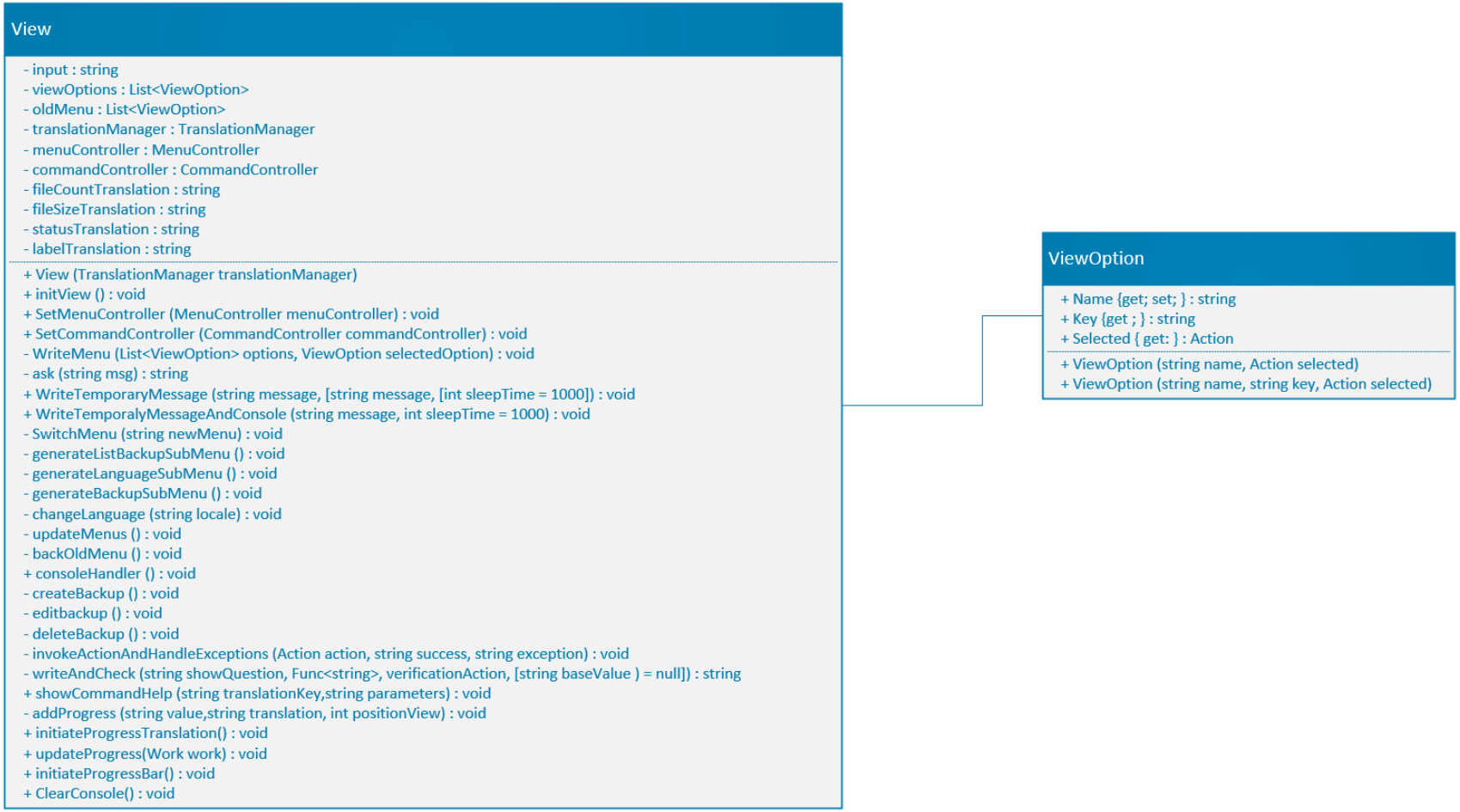
A la suite de cela, nous avons réaliser des diagramme de classes afin de visualiser quels sont les différentes classes que nous avons utiliser afin de réaliser notre interface.

1) Controller

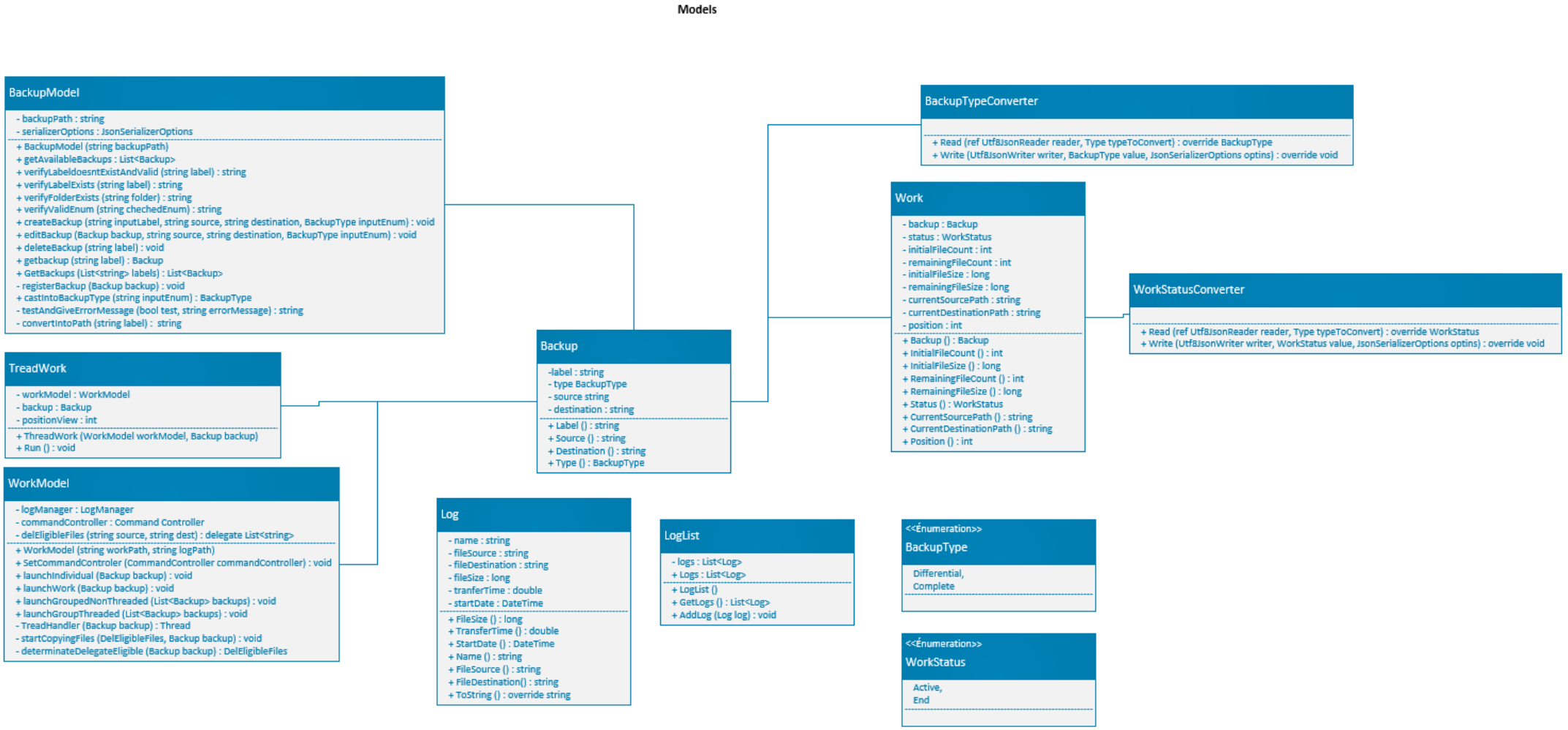


2) View

View

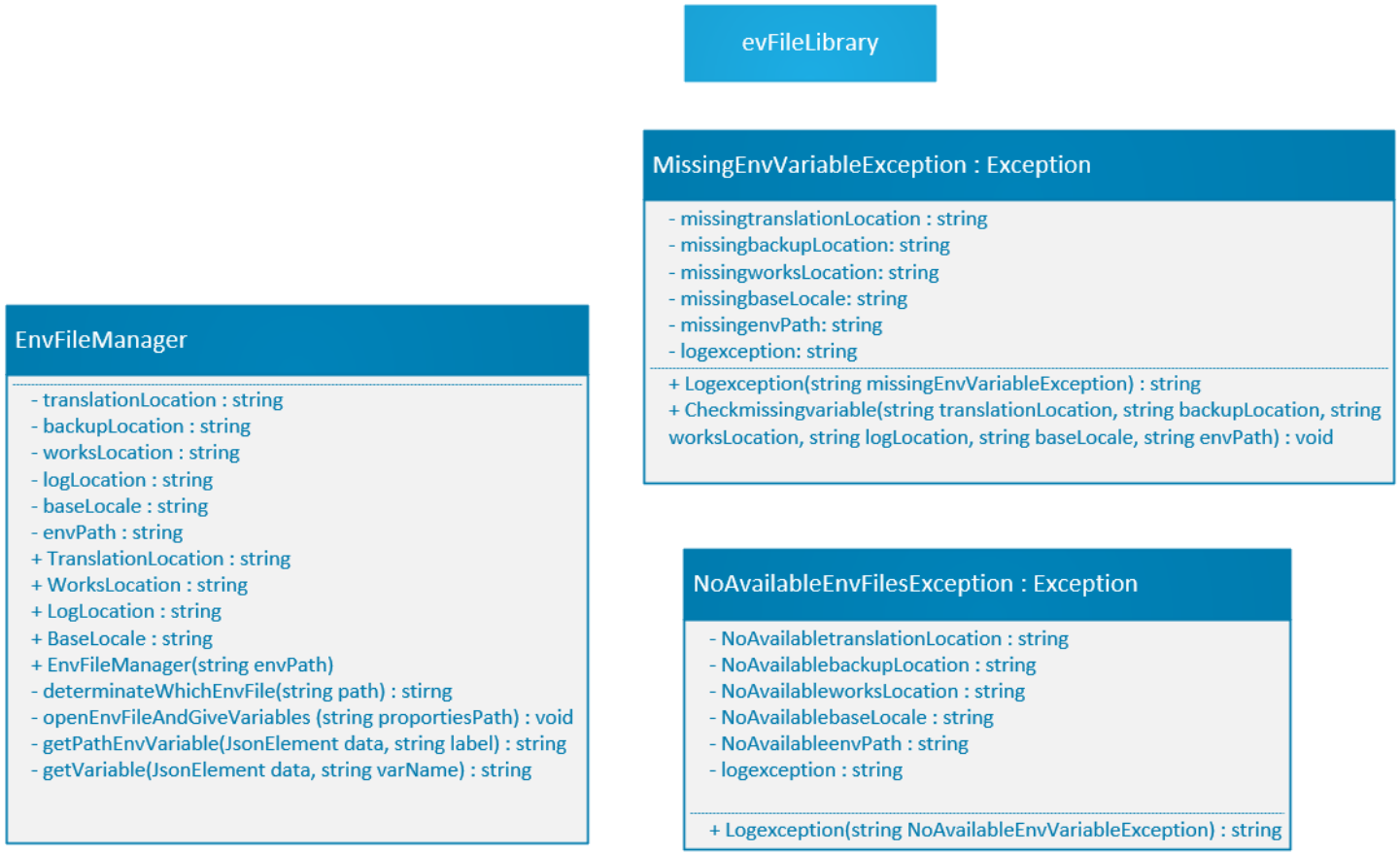

View
- input : string
- viewOptions : List<ViewOption>
- oldMenu : List<ViewOption>
- translationManager : TranslationManager
- menuController : MenuController
- commandController : CommandController
- fileCountTranslation : string
- fileSizeTranslation : string
- statusTranslation : string
- labelTranslation : string
+ View (TranslationManager translationManager)
+ initView () : void
+ SetMenuController (MenuController menuController) : void
+ SetCommandController (CommandController commandController) : void
- WriteMenu (List<ViewOption> options, ViewOption selectedOption) : void
- ask (string msg) : string
+ WriteTemporaryMessage (string message, [string message, [int sleepTime = 1000]) : void
+ WriteTemporalyMessageAndConsole (string message, int sleepTime = 1000) : void
- SwitchMenu (string newMenu) : void
- generateListBackupSubMenu () : void
- generateLanguageSubMenu () : void
- generateBackupSubMenu () : void
- changeLanguage (string locale) : void
- updateMenus () : void
- backOldMenu () : void
+ consoleHandler () : void
- createBackup () : void
- editbackup () : void
- deleteBackup () : void
- invokeActionAndHandleExceptions (Action action, string success, string exception) : void
- writeAndCheck (string showQuestion, Func<string>, verificationAction, [string baseValue) = null]) : string
+ showCommandHelp (string translationKey,string parameters) : void
- addProgress (string value,string translation, int positionView) : void
+ initiateProgressTranslation() : void
+ updateProgress(Work work) : void
+ initiateProgressBar() : void
+ ClearConsole() : void
ViewOption
+ Name {get; set; } : string
+ Key {get ; } : string
+ Selected { get: } : Action
+ ViewOption (string name, Action selected)
+ ViewOption (string name, string key, Action selected)

3) Model


Models
BackupModel
- backupPath : string
- serializerOptions : JsonSerializerOptions
+ BackupModel (string backupPath)
+ getAvailableBackups : List<Backup>
+ verifyLabeldoesntExistAndValid (string label) : string
+ verifyLabelExists (string label) : string
+ verifyFolderExists (string folder) : string
+ verifyValidEnum (string chechedEnum) : string
+ createBackup (string inputLabel, string source, string destination, BackupType inputEnum) : void
+ editBackup (Backup backup, string source, string destination, BackupType inputEnum) : void
+ deleteBackup (string label) : void
+ getbackup (string label) : Backup
+ GetBackups (List<string> labels) : List<Backup>
- registerBackup (Backup backup) : void
+ castIntoBackupType (string inputEnum) : BackupType
- testAndGiveErrorMessage (bool test, string errorMessage) : string
- convertIntoPath (string label) : string
BackupTypeConverter
+ Read (ref Utf8JsonReader reader, Type typeToConvert) : override BackupType
+ Write (Utf8JsonWriter writer, BackupType value, JsonSerializerOptions optins) : override void
Work
- backup : Backup
- status : WorkStatus
- initialFileCount : int
- remainingFileCount : int
- initialFileSize : long
- remainingFileSize : long
- currentSourcePath : string
- currentDestinationPath : string
- position : int
+ Backup () : Backup
+ InitialFileCount () : int
+ InitialFileSize () : long
+ RemainingFileCount () : int
+ RemainingFileSize () : long
+ Status () : WorkStatus
+ CurrentSourcePath () : string
+ CurrentDestinationPath () : string
+ Position () : int
WorkStatusConverter
+ Read (ref Utf8JsonReader reader, Type typeToConvert) : override WorkStatus
+ Write (Utf8JsonWriter writer, WorkStatus value, JsonSerializerOptions optins) : override void
TreadWork
- workModel : WorkModel
- backup : Backup
- positionView : int
+ ThreadWork (WorkModel workModel, Backup backup)
+ Run () : void
Backup
-label : string
- type BackupType
- source string
- destination : string
+ Label () : string
+ Source () : string
+ Destination () : string
+ Type () : BackupType
WorkModel
- logManager : LogManager
- commandController : Command Controller
- delEligibleFiles (string source, string dest) : delegate List<string>
+ WorkModel (string workPath, string logPath)
+ SetCommandControler (CommandController commandController) : void
+ launchIndividual (Backup backup) : void
+ launchWork (Backup backup) : void
+ launchGroupedNonThreaded (List<Backup> backups) : void
+ launchGroupThreaded (List<Backup> backups) : void
- TreadHandler (Backup backup) : Thread
- startCopyingFiles (DelEligibleFiles, Backup backup) : void
- determinateDelegateEligible (Backup backup) : DelEligibleFiles
Log
- name : string
- fileSource : string
- fileDestination : string
- fileSize : long
- tranferTime : double
- startDate : DateTime
+ FileSize () : long
+ TransferTime () : double
+ StartDate () : DateTime
+ Name () : string
+ FileSource () : string
+ FileDestination() : string
+ ToString () : override string
LogList
- logs : List<Log>
+ Logs : List<Log>
+ LogList ()
+ GetLogs () : List<Log>
+ AddLog (Log log) : void
<<Énumeration>>
BackupType
Differential,
Complete
<<Énumeration>>
WorkStatus
Active,
End

4) Librairies

evFileLibrairy

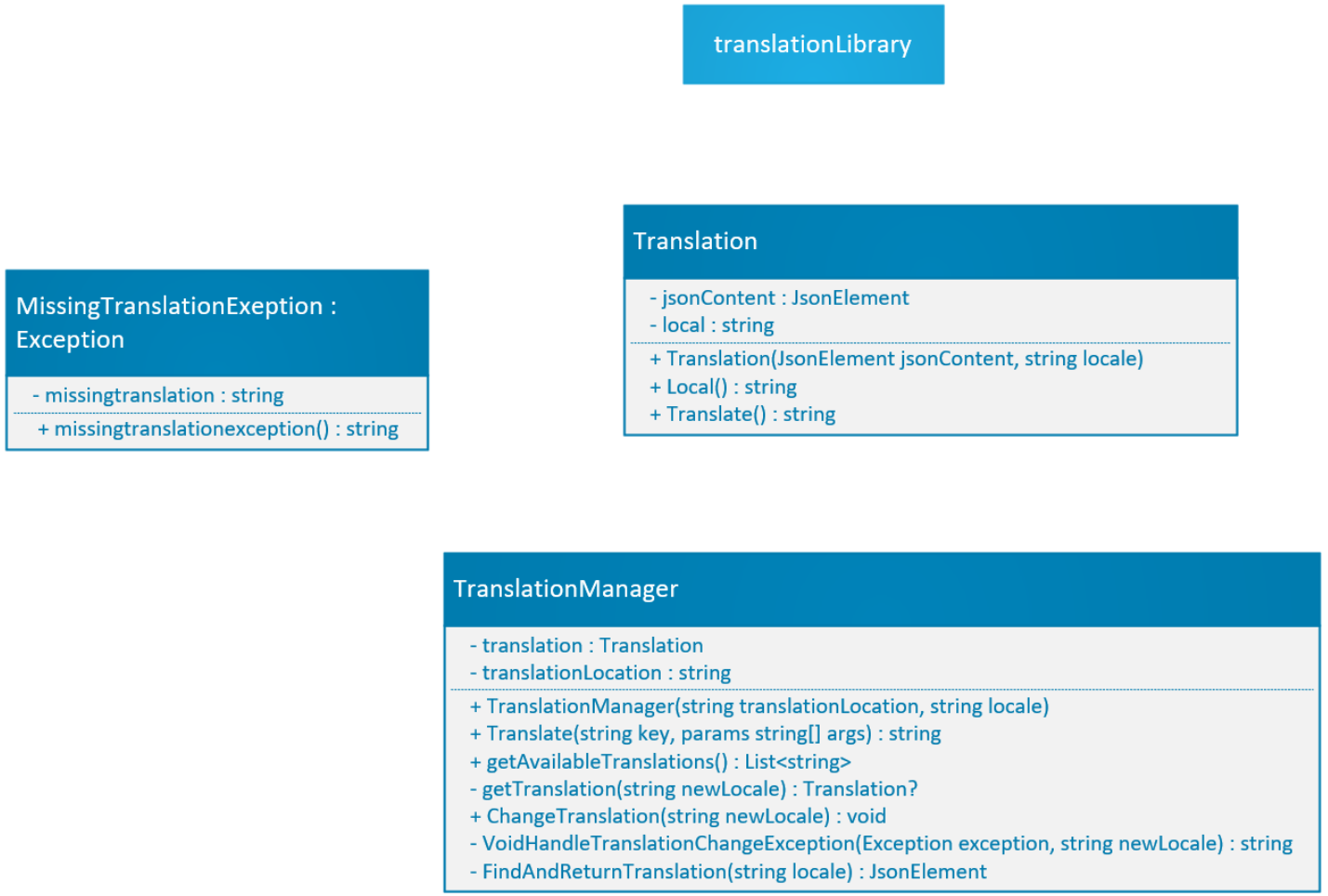


FileLibrary

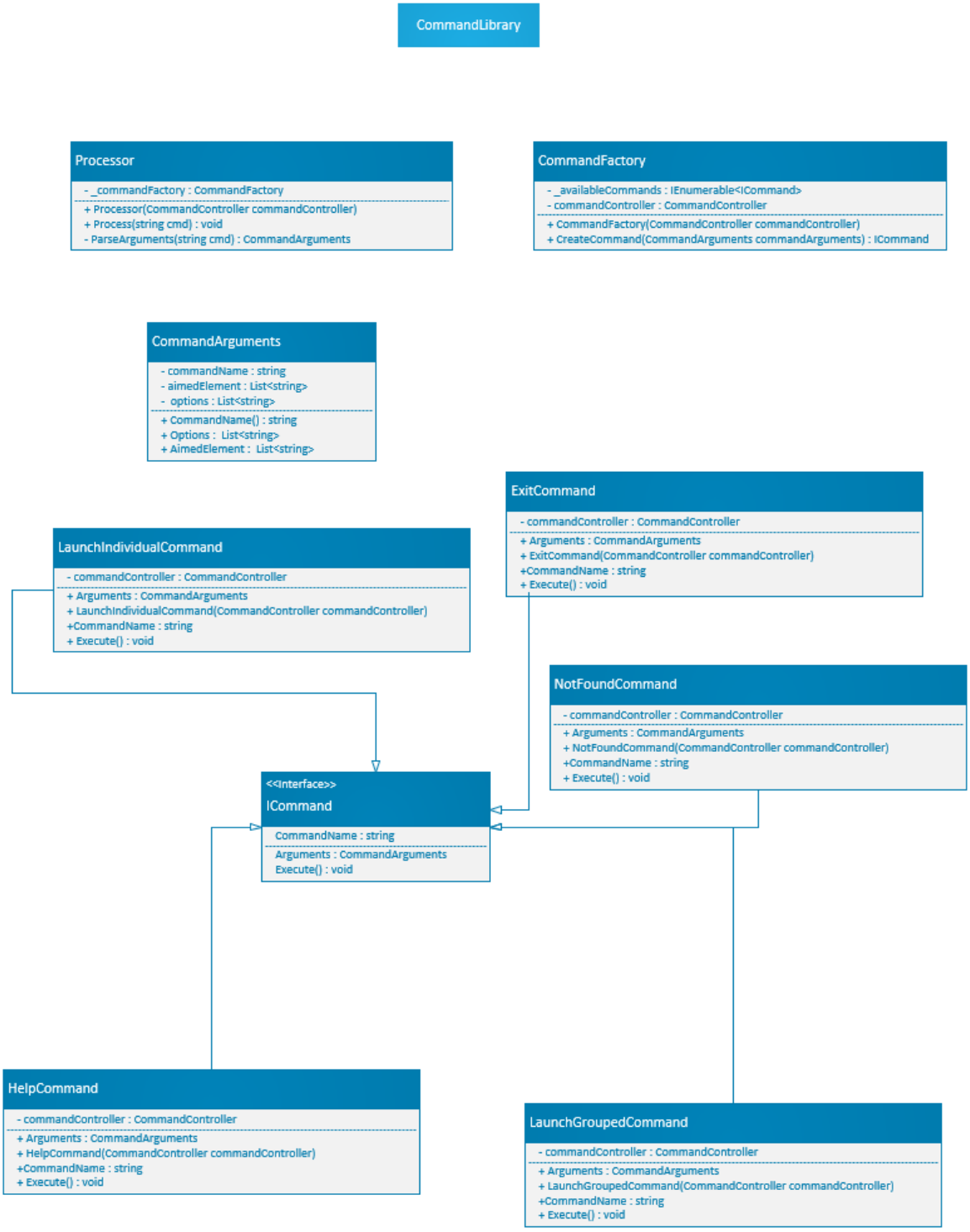
LogLibrary


LogLibrary
WorkLoggingThread
- _workFileQueue : BlockingCollection<Log>
- logManager : LogManager
- timestamp : string
+ WorkLoggingThread(BlockingCollection<Work> workFileQueue, LogManager logManager, string timestamp)
+ Run() : void
LogManager
- WorkFileName : string
- logPath : string
- workPath : string
- logFileName : string
- logFileQueue : BlockingCollection<Log>
- workFileQueue : BlockingCollection<Work>
- loggingThreadHandle : Thread
- workingThreadHandle : Thread
- serializerOptions : JsonSerializerOptions
+ LogManager(string workPath,string logPath)
+ initiateLogging() : void
+ stopLogging() : void
- setAndCreateLogFile() : void
- setAndCreateWorkFile() : void
- initiateWorkFile() : string
- initiateLogFile() : string
- SerializeLogList(LogList logList) : string
- SerializeWorkList(Dictionary<string, List<Work>> workList) : string
+ appendLogToLogList(Log log) : void
+ appendWorkToWorkList(Work work) : void
+ updateLogFile(LogList logList) : void
+ updatedWorkLogFile(Dictionary<string, List<Work>> workList) : void
+ getLogList() : LogList
+ getWorkList() : Dictionary<string, List<Work>>
FileLoggingThread
- _logFileQueue : BlockingCollection<Log>
- logManager : LogManager
+ FileLoggingThread(BlockingCollection<Log> logFileQueue, LogManager logManager)
+ Run() : void

TranslationLibrary



CommandLibrary



II) Documentation

A) Menu

Au lancement de l'application, on se trouve sur un menu sur lequel on peut naviguer avec les flèches, et entrée pour accéder aux différentes fonctionnalités de l'application.

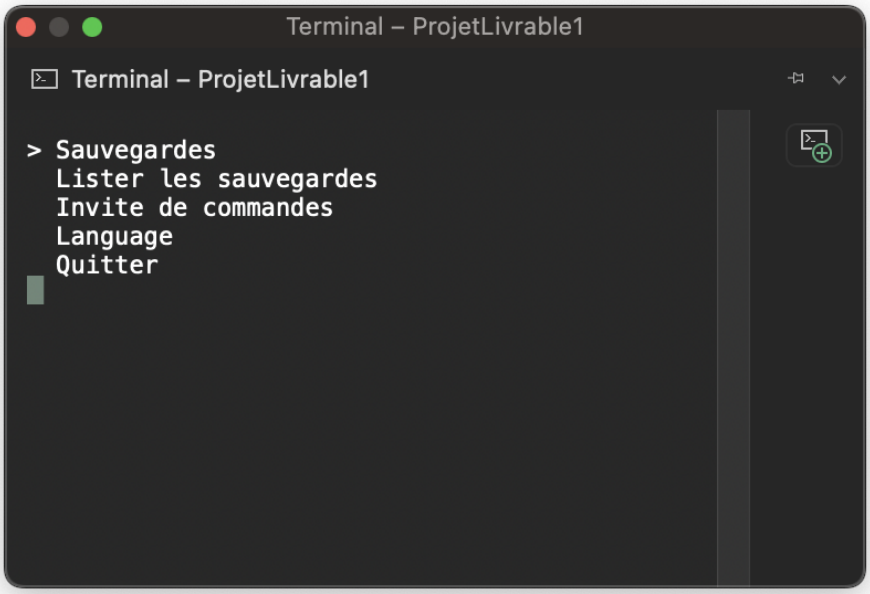


Figure 1 : Menu

B) Sauvegarde

Ici de la même manière que dans le menu on sélectionne l'opération que l'on souhaite effectuer.

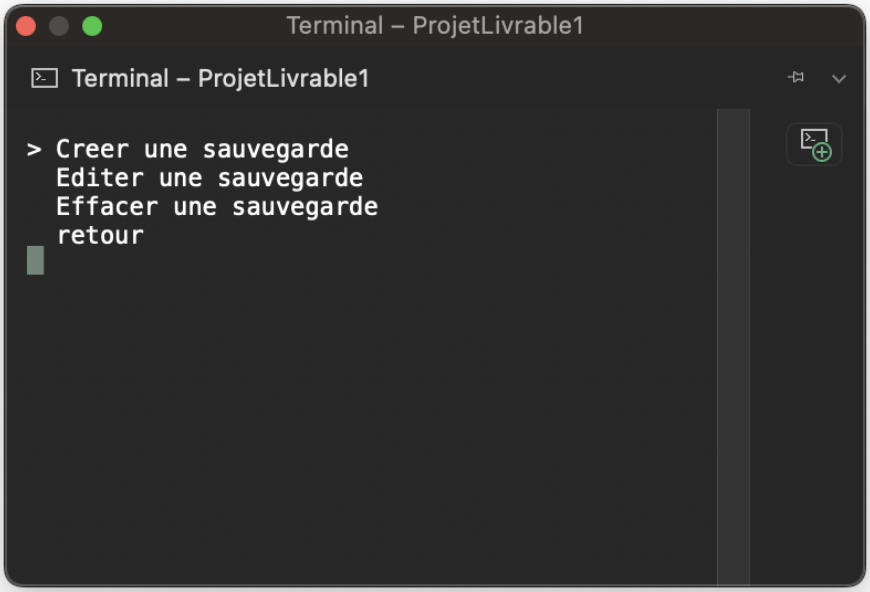


Figure 2 : Sauvegarde

1) Créer une sauvegarde

Dans on donne le nom du fichier que l'on veut sauvegarder.

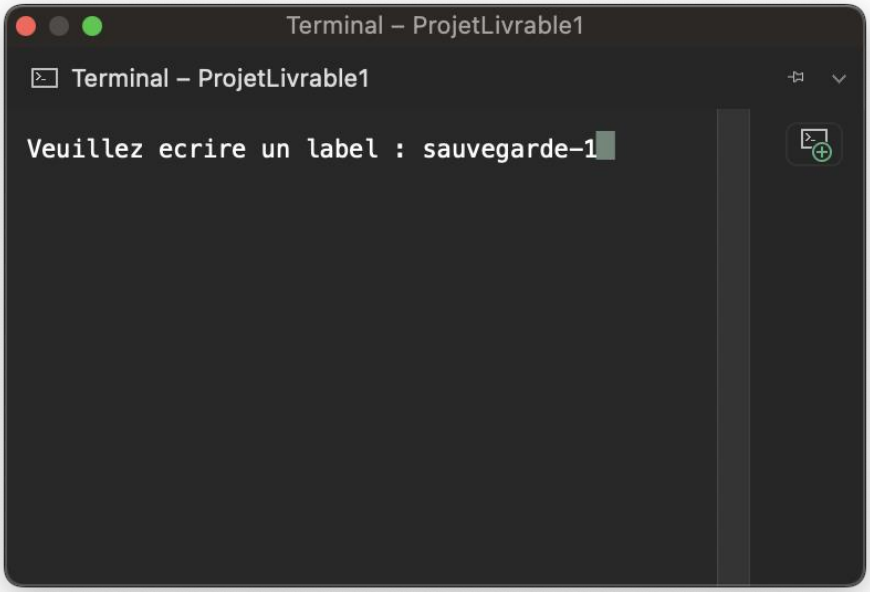


Figure 3 : Label

Puis on indique le chemin du dossier ou se trouve le fichier à sauvegarder

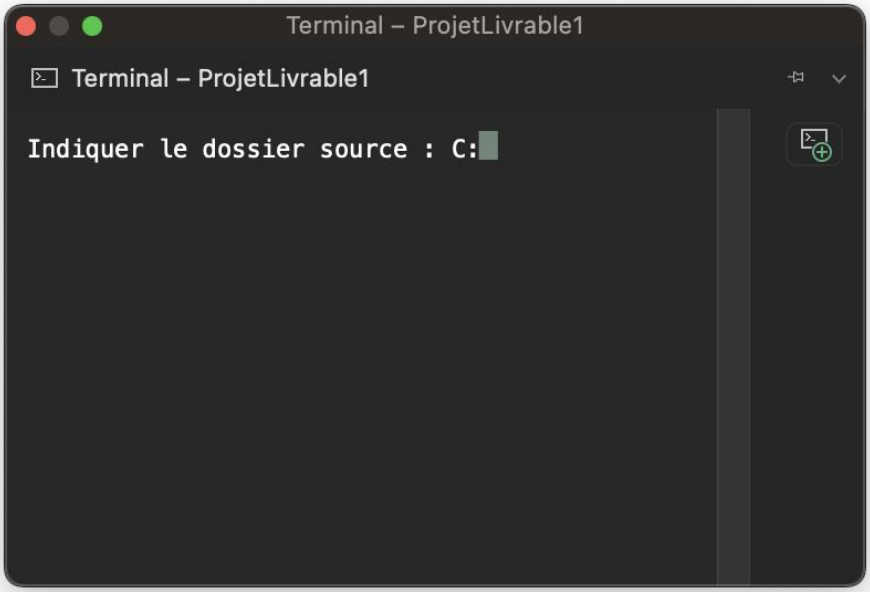


Figure 4 : Dossier source

Et le chemin du dossier ou l'on veut que la sauvegarde se créer

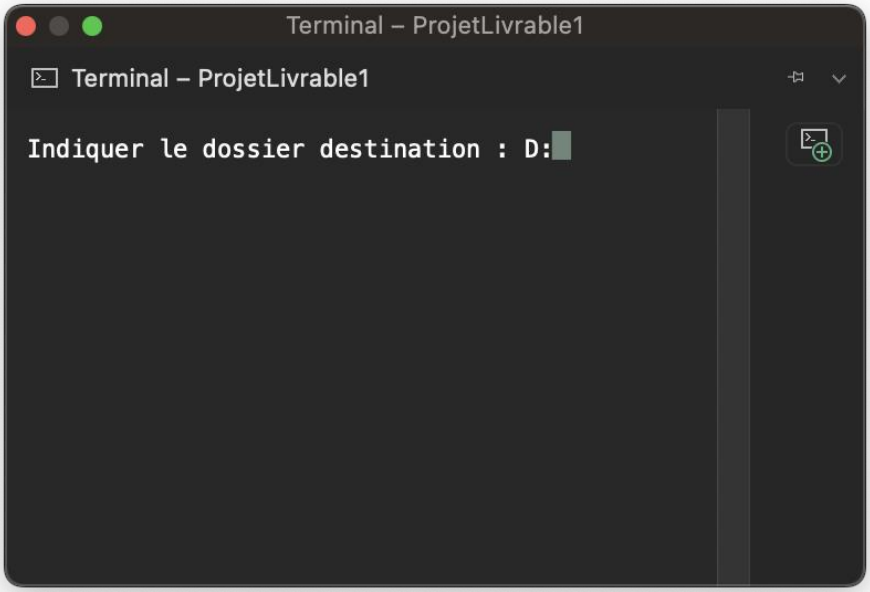


Figure 5 : Dossier destination

Et enfin on écrit ici le type de la sauvegarder

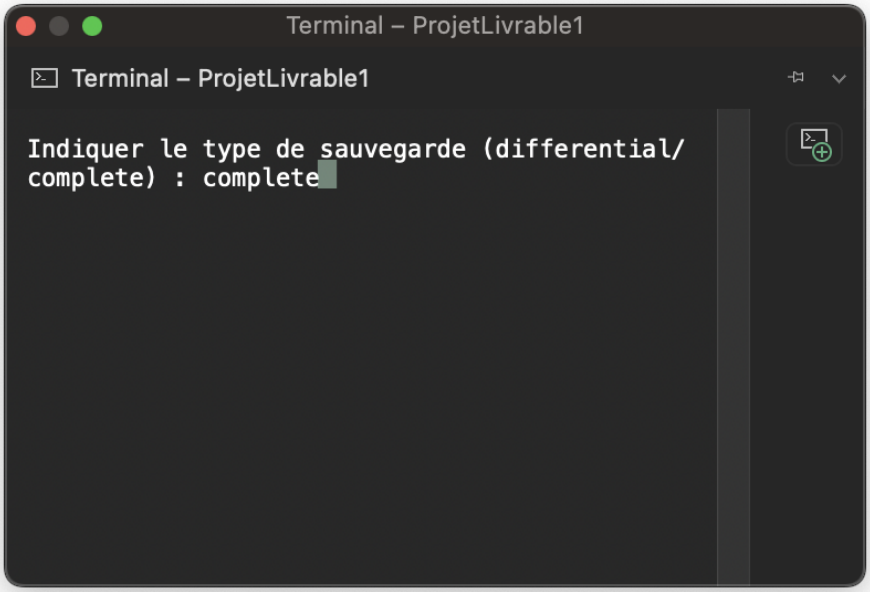


Figure 6 : Type de sauvegarde

2) Editer une sauvegarde

électionne en indiquer le label de cette derniere

Figure 7 : Editer une sauvegarde

Ensuite l'application va proposer de changer les différents paramètres, si on ne veut pas changer un paramètre, on laissera le champs vide.

Exemple : ici on ne change pas

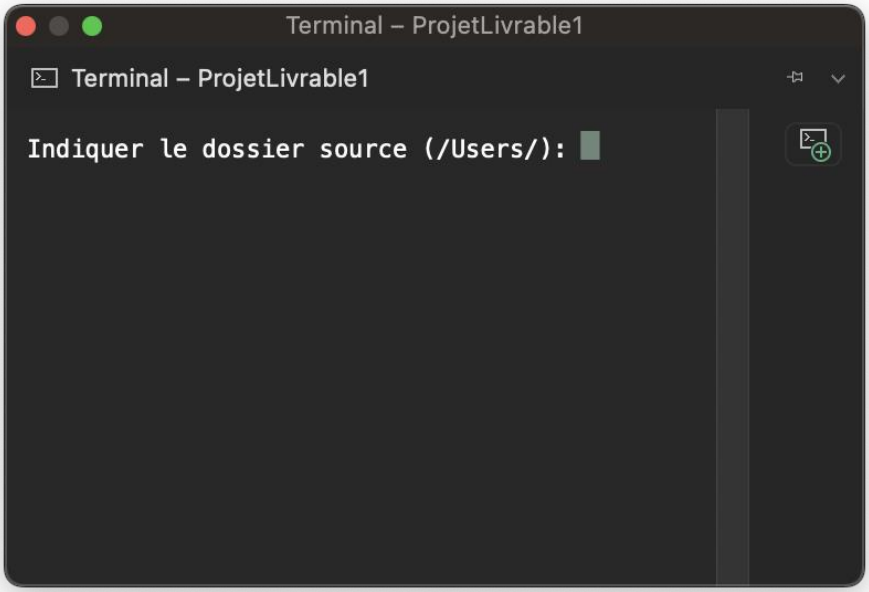


Figure 8 : Modification dossier source

Alors que ici on le change.

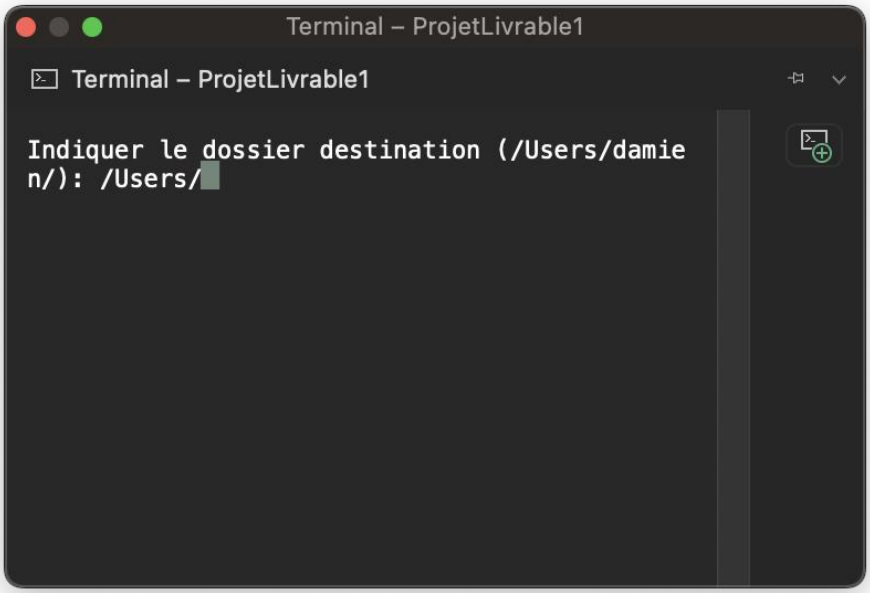


Figure 9 : Modification dossier destination

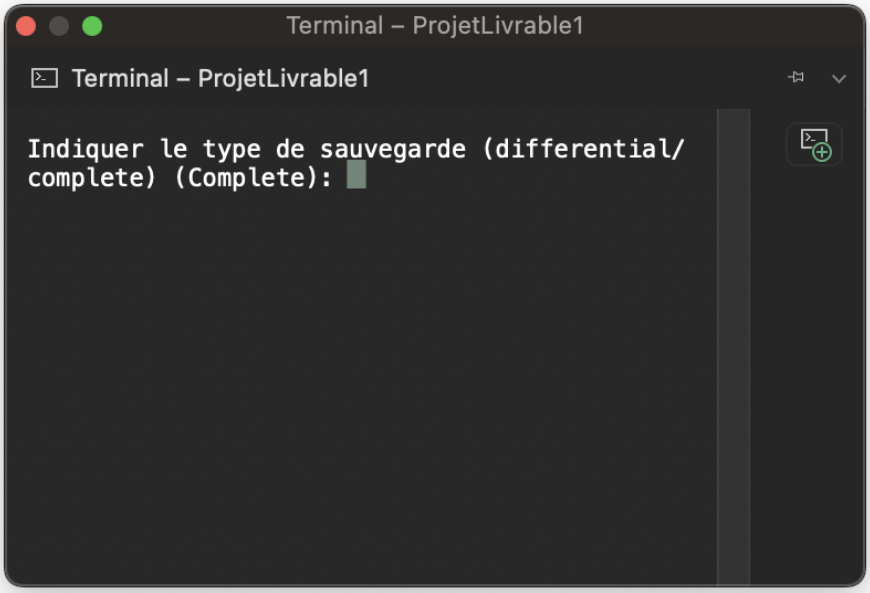


Figure 10 : Modification type de sauvegarde

3) Effacer une sauvegarde

Pour effacer une sauvegarde, il suffit d'indiquer son nom

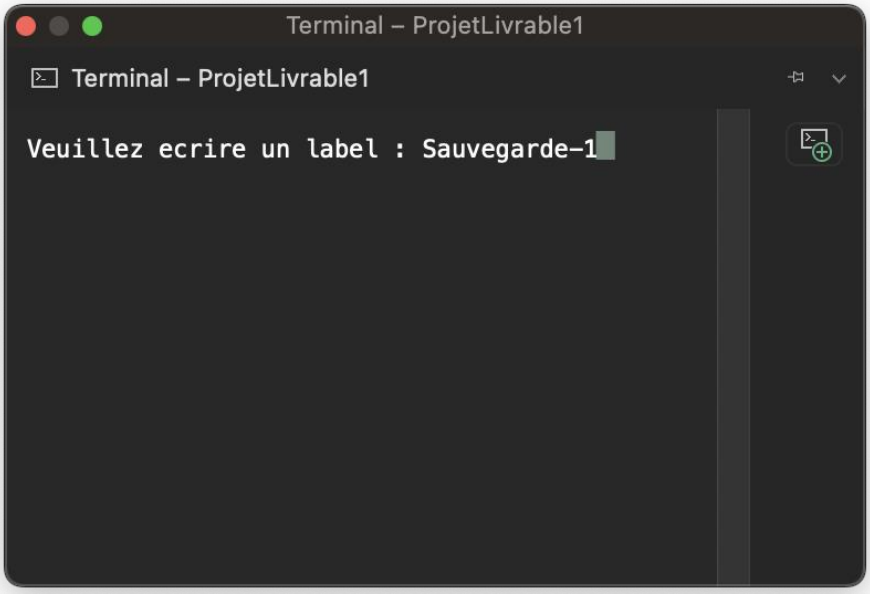


Figure 11 : Effacer une sauvegarde

C) Lister les sauvegardes

En choisissant le menu « Lister les sauvegardes » on obtient la liste des sauvegardes.

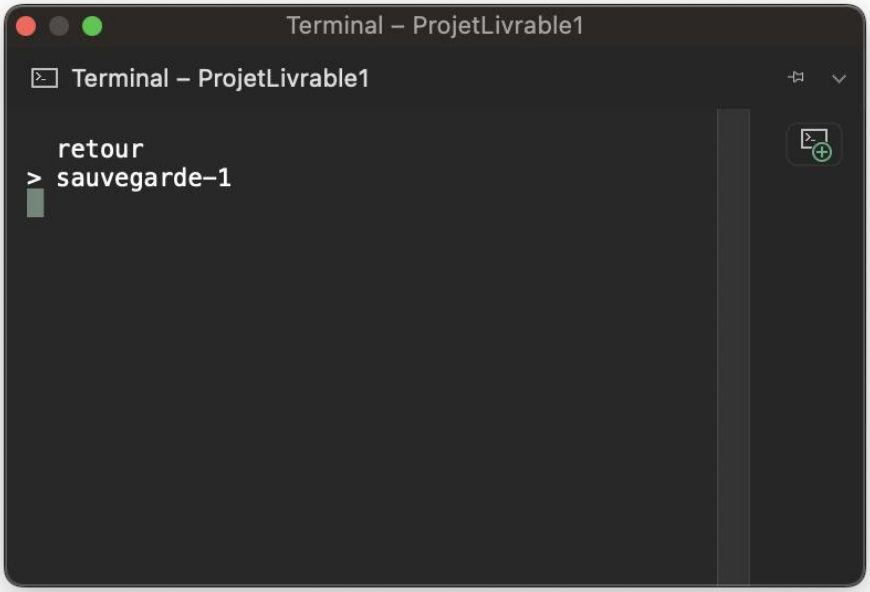


Figure 12 :Lister les sauvegardes

En cliquant sur une sauvegarde on obtient plus de détail

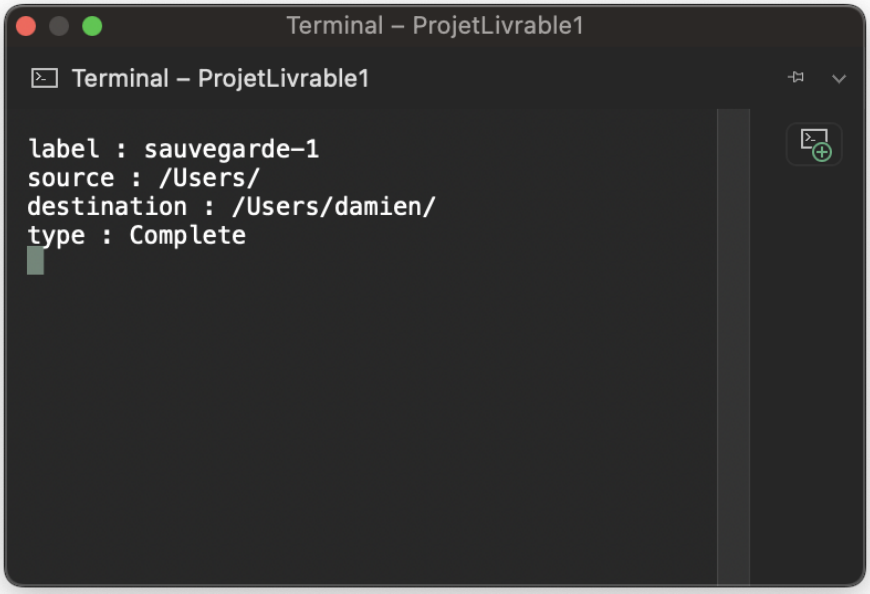


Figure 13 : Information sauvegarde

D) Langages

Enfin on peut changer la langue dans le menu « Language.

Figure 14 : Langages

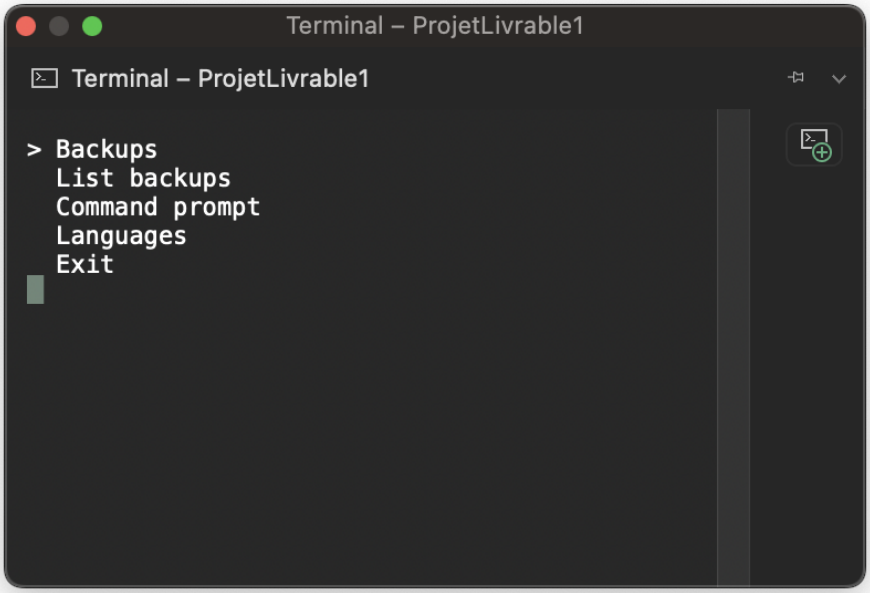


Figure 15 : English menu

Conclusion

Pour conclure nous avons réaliser les différents diagrammes UML pour conceptualiser nôtre projet. Puis nous avons codé l'application console. Et enfin nous avons rédiger une documentation utilisateur.