CESI
ÉCOLE D'INGÉNIEURS

DOCUMENT TECHNIQUE – EasySave

Version 1.0

GROUPE 2:

- FABO GAETAN
- ATONLE EDSON
- BOGNING JICELIN
- KINDA ALEX
- FOGOU JORDAN

Table des matières

1. Introduction

Ce document technique décrit l'architecture, les composants, les formats de données et les flux critiques de l'application EasySave v1.0.

2. Architecture générale

EasySave adopte une architecture MVC légère (mais pas exactement):

- **Model** : gestion des données métier, configurations et opérations de sauvegarde.

- **View** : affichage console multilingue (English/French).

- **Controller** : orchestration de l'application, parsing CLI, boucle interactive.

- **Logging DLL** : journalisation quotidienne des transferts de fichiers.

3. Composants

3.1 Model

Fichier : **Model.cs**

Responsabilités :

- Chargement et enregistrement de backupList.json (jobs de sauvegarde).

- Gestion de l'état temps réel dans state.json.

- Exécution des sauvegardes : CompleteSave() et DifferentialSave().

- Appel à la journalisation via EasySoft.Logging.DailyLogger.

3.2 View

Fichier : **View.cs**

Responsabilités :

- Affichage multilingue en console (choix de la langue, menus, invites).

- Méthodes ShowMenu(), ShowSubMenu(), ShowResource(), etc.

3.3 Controller

Fichier : **Controller.cs**

Responsabilités :

- Parsing des arguments CLI (1-3 ou 1;3).

- Boucle interactive : traitement des options 1,2,3.

- Appel au Model pour créer ou exécuter des sauvegardes.

- Gestion des pauses, messages de succès et retour au menu.

3.4 EasySoft.Logging

Projet : **EasySoft.Logging**

Responsabilités :

- Classe DailyLogger pour journaliser chaque transfert de fichier.

- Format JSON journalier : DailyLogs\YYYY-MM-DD.json.

- Implémentation isolée pour réutilisation future.

4. Formats de données

4.1 backupList.json

Contient un tableau d'objets Backup :

- **ResourceBackup** : chemin source

- **TargetBackup** : chemin cible

- **SaveName** : identifiant unique

- **Type** : full / differential

- **MirrorBackup** : chemin du dernier full pour diff

4.2 state.json

Contient un tableau d'objets DataState :

- SaveNameState, BackupDateState, SourceFileState, TargetFileState

- TotalFileState, TotalSizeState, FileRestState, TotalSizeRestState, ProgressState

- Mis à jour en temps réel pour suivi.

4.3 DailyLogs/YYYY-MM-DD.json

Contient un tableau de FileTransferLogEntry :

- Name, FileSource, FileTarget, FileSize, FileTransferTime, Time

- Ajout d'une entrée JSON par fichier copié.

5. Flux d'exécution

1. Lancement (CLI ou interactif).

2. Choix de la langue via View.ShowLanguageSelection().

3. Affichage du menu principal.

4. En fonction de l'option :

  - OpenBackup() → Appel Model.LoadSave() → copy → logging → message succès.

  - CreateBackup() → collecte paramètres → Model.AddSave() → mise à jour backupList.json.

  - ListBackups() → affichage filtré.

5. Boucle jusqu'à 0 pour quitter.

6. Gestion des erreurs

- Utilisation de TryParse et validation de chemins.

- Retour à l'écran principal sans crash.

- Messages clairs pour chemins incorrects et jobs introuvables.

- Indices hors plage en CLI sont signalés.

7. Internationalisation

Encapsulée dans View.SelectedLanguage. Toutes les chaînes sont gérées en fonction de la langue choisie.

8. Build et déploiement

Commande : dotnet build

Configuration : net8.0, Nullable enabled, implicit usings.

Fichiers à versionner : Controller/, Model/, View/, EasySoft.Logging/, *.csproj, README, .gitignore.

Ne pas versionner : bin/, obj/, Works/, State/, DailyLogs/ (générés).