## Documentation développeur

### Tim Vinciguerra

### Ethan Jouvin

### Armand

Table des matières

[Documentation développeur 1](#_Toc190119754)

[1. Structure du projet 2](#_Toc190119755)

[2. Exécution du projet 2](#_Toc190119756)

[3. Emplacement des fichiers de log et d’état 3](#_Toc190119757)

[4. Lien entre les différentes parties de l’application 3](#_Toc190119758)

## Structure du projet

Notre application est composée en trois parties :

* Easy CLI (contient le main qui est exécuté en CLI peut être considéré comme la view)
* EasyLib (Contient le cœur de notre application et toute la logique métier contient quant à lui le model, le viewModel ainsi que les services)
* EasyLog (Contient les models ainsi que les services pour les fichiers Log et d’état)

Notre application est ainsi extrêmement segmentée augmentant ainsi la simplicité de changement d’interface par exemple.

## Exécution du projet

Afin d’exécuter notre projet il faut posséder une version de dotnet qui est la 9.0

Pour lancer l’exécution lancez les commandes suivantes :

-cd .\EasyCLI\ (afin de vérifier que nous nous situons bien dans le CLI c’est-à-dire le projet a exécuter)

-dotnet build (pour builder le .exe)

-dotnet run (pour l’executer)

-ensuite nous devons choisir la langue (en/fr)

L’application va ensuite se lancer.

## Emplacement des fichiers de log et d’état

Notre application possède son propre système de log et d’état ainsi lors de la première action effectuée en général une création de backup un dossier sera créer automatiquement dans le disque C:\ du nom de Logs



Il contiendra tous les fichiers nécessaires aux logsUne image contenant texte, capture d’écran, Police, ligne

Le contenu généré par l’IA peut être incorrect.

## Lien entre les différentes parties de l’application

Dans notre application les liens sont les suivants :

EasyCli va s’exécuter au démarrage de l’application (fichier Program.cs) : il va s’occuper de l’affichage et appeler ---------------------------------------------------------------------------------------------

-> EasyLib, plus précisément il va appeler BackupViewModel.cs qui va s’occuper de paramétrer les différentes actions entrées par l’utilisateur et en fonction de cela appeler -------------------------> Toujours dans EasyLib, BackupServices.cs qui contient toute la logique métier des différentes actions et ce fichier va appeler---------------------------------------------------------------------> Les Services des fichiers de logs et d’états ainsi que le model Backupmodel.cs de EasyLib

--------------------------------------------------------------------------------------------------------------------------> Les services des fichiers Logs vont pour finir appeler leurs model respectifs.