Livrable 1

**EasySave - Version 1.0**

**Livrable 1 - Document de Spécification et d’Analyse**

**Réalisé par le Groupe 2**

* Sahraoui Abderahmane
* Guichi Safi
* Ouandalous Mohamed
* Skander Wassim
  1. **Sommaire :**

1. Page de garde
2. Sommaire
3. Introduction
4. Objectifs du projet
5. Définition détaillée des diagrammes UML
6. Présentation de la logique du modèle MVC appliquée à EasySave 1.0
7. Description détaillée de l’utilisation de l’application EasySave 1.0
8. Conclusion
9. **Introduction**

Le volume croissant des données numériques et la dépendance des organisations à l’égard de l’information numérique rendent la gestion des sauvegardes de plus en plus stratégique. La perte de données peut avoir des conséquences importantes, tant sur le plan technique qu’économique. Dans ce contexte, il devient indispensable de mettre en place des solutions fiables, simples d’utilisation et adaptées aux besoins des utilisateurs.

C’est dans cette optique qu’a été conçu **Économie facile**, un logiciel de sauvegarde permettant de sécuriser les données de manière efficace à travers une interface en ligne de commande. EasySave a été pensé pour répondre à des besoins réels en matière de gestion de sauvegardes locales, sur disques externes ou sur lecteurs réseaux, tout en restant accessible à tous types d’utilisateurs.

La version 1.0 d’EasySave propose un ensemble de fonctionnalités fondamentales telles que la création de travaux de sauvegarde, leur exécution, la gestion du suivi en temps réel et l’enregistrement des actions dans des fichiers journaux structurés au format JSON. Le logiciel est conçu pour fonctionner aussi bien en français qu’en anglais afin de répondre à une diversité d’utilisateurs.

Dans ce document, nous présentons le cadre de ce premier livrable, ses objectifs, ainsi que l’approche méthodologique adoptée. Nous détaillerons les choix d’architecture logicielle, en particulier l’utilisation du modèle MVC, les mécanismes d’interaction avec l’application, les diagrammes UML produits et enfin les outils mis en œuvre pour le suivi du projet.

1. **Objectifs du projet**

Le projet **Sauvegarde facile 1.0** a pour principal objectif de proposer une **Solution simple, robuste et adaptable** pour la **sauvegarde automatisée de données**. Il vise à répondre à des besoins courants rencontrés par les utilisateurs souhaitant protéger leurs fichiers contre la perte accidentelle, les erreurs humaines ou les défaillances matérielles.

**Les objectifs techniques de la version 1.0 sont les suivants :**

* **Permettre la création et la gestion de plusieurs travaux de sauvegarde personnalisés**, chacun défini par un nom, un dossier source, un dossier de destination et un type de sauvegarde (complète ou différentielle).
* **Proposer une exécution flexible des sauvegardes**, soit de manière unitaire, soit en séquence, en permettant de sélectionner les travaux à exécuter via une commande simple en ligne de commande.
* **Garantir la traçabilité des actions réalisées** par la génération automatique d’un **fichier journalier (log)** qui enregistre toutes les opérations effectuées : copie de fichiers, erreurs, durées d’exécution, chemins des fichiers, etc.
* **Assurer le suivi en temps réel de l’état des travaux en cours**, grâce à la production d’un **fichier d’état dynamique** qui indique la progression, les fichiers en cours de traitement, et les statistiques d’avancement.
* **Offrir une prise en charge multilingue (français et anglais)**, permettant à tout utilisateur d’utiliser le logiciel dans la langue de son choix.
* **Respecter les bonnes pratiques de conception logicielle**, notamment à travers l’architecture **Modèle - Vue - Contrôleur (MVC)**, garantissant la clarté, la maintenabilité et la possibilité d’évolution vers une future interface graphique en version 2.0.
* **Produire des fichiers de suivi au format JSON**, facilement lisibles et exploitables par d’autres outils, avec une mise en forme adaptée à une consultation humaine.

Ce premier livrable constitue une **base solide** sur laquelle des évolutions futures pourront être envisagées, notamment l’intégration d’une interface graphique et l’ajout de fonctionnalités avancées.

1. **Définition détaillée des diagrammes UML**

Afin de modéliser les fonctionnalités, les interactions et la structure interne du logiciel **EasySave 1.0**, plusieurs **diagrammes UML** ont été réalisés. Ces diagrammes facilitent la compréhension du comportement du système et la communication entre les membres de l’équipe de développement.

**1. Diagramme de cas d’utilisation (Use Case)**

Le diagramme de cas d’utilisation présente une **vue fonctionnelle** du système du point de vue de l’utilisateur. Il décrit les **principales actions** que l’utilisateur peut effectuer en interaction avec le logiciel.

**Cas d’utilisation identifiés :**

* **Créer un travail de sauvegarde** : Permet à l’utilisateur de configurer une nouvelle sauvegarde avec un nom, une source, une destination et un type.
* **Modifier un travail existant** : Offre la possibilité de mettre à jour les paramètres d’un travail.
* **Supprimer un travail** : Supprime définitivement un travail de la liste.
* **Lister les travaux enregistrés** : Affiche les informations des travaux configurés.
* **Exécuter un ou plusieurs travaux** : Lance l’exécution de sauvegardes choisies par l’utilisateur.
* **Exécuter tous les travaux** : Lance l’exécution séquentielle de toutes les sauvegardes existantes.
* **Changer la langue d’utilisation** : Permet de basculer entre le français et l’anglais.
* **Quitter l’application** : Ferme l’application proprement.

Ces cas d’utilisation traduisent les **besoins fonctionnels** exprimés dans le cahier des charges et guident le développement de l’interface utilisateur.

Une image contenant diagramme, texte, capture d’écran, Plan

Le contenu généré par l’IA peut être incorrect.

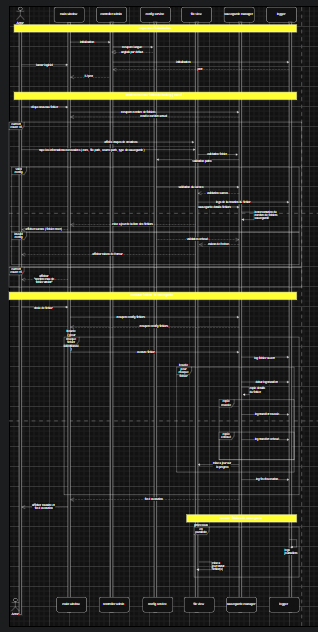
**2. Diagramme de séquence**

Le diagramme de séquence détaille la **chronologie des échanges** entre l’utilisateur et les différentes **composantes logicielles** lorsque l’utilisateur interagit avec l’application.

**Scénario principal : exécution d’un travail de sauvegarde**

1. L’utilisateur sélectionne l’exécution d’un travail.
2. Le **Contrôleur principal** interprète la commande.
3. Le contrôleur interroge le **gestionnaire de travaux** pour charger les paramètres du travail sélectionné.
4. Le contrôleur déclenche la sauvegarde en appelant la **logique métier** qui :
   * Copie les fichiers du répertoire source vers le répertoire de destination.
   * Met à jour en temps réel le **fichier d’état**.
   * Enregistre les actions dans le **fichier log journalier**.
5. Le contrôleur informe l’utilisateur de la fin du processus.

Ce diagramme illustre le **flux d’information** et le **rôle central du contrôleur**, garantissant que la vue et le modèle interagissent uniquement par son intermédiaire.



**3. Diagramme d’activité**

Le diagramme d’activité décrit le **flux de travail de l’utilisateur** lorsqu’il utilise l’application. Il permet de visualiser les **décisions**, les **boucles**, et les **transitions entre les différentes étapes**.

**Étapes principales :**

1. Lancement de l’application et affichage du menu principal.
2. L’utilisateur choisit une action (créer, modifier, supprimer, lister, exécuter, changer de langue ou quitter).
3. Le système vérifie les saisies et affiche les résultats ou les messages d’erreur.
4. Si l’utilisateur lance une sauvegarde, le système traite les fichiers, met à jour le fichier d’état et le fichier log.
5. Le menu est de nouveau affiché jusqu’à ce que l’utilisateur décide de quitter.

Ce diagramme montre que l’application fonctionne en **cycle continu** jusqu’à ce que l’utilisateur décide de **fermer le programme**.

Une image contenant texte, capture d’écran, diagramme, circuit

Le contenu généré par l’IA peut être incorrect.

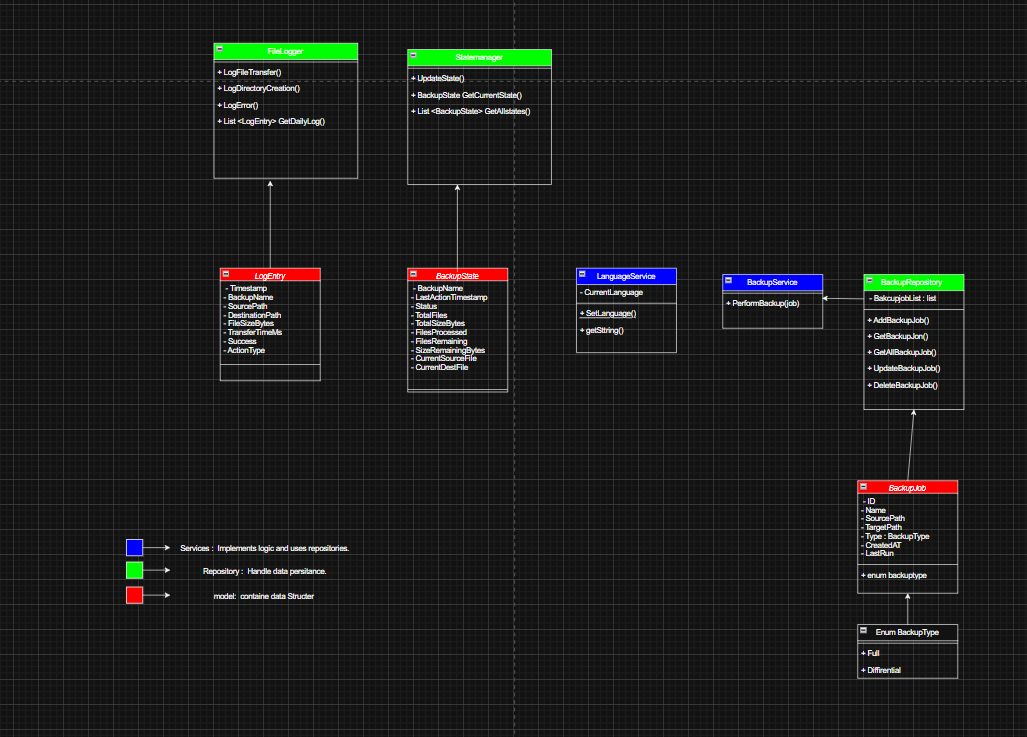
**4. Diagramme de classes**

Le diagramme de classes présente la **structure interne** du logiciel en identifiant les **classes principales**, leurs **attributs**, leurs **méthodes**, et les **relations entre elles**.

**Principales classes identifiées :**

* **SaveWork** : Représente un travail de sauvegarde avec ses propriétés (nom, source, destination, type).
* **SaveWorkManager** : Gère l’ensemble des travaux de sauvegarde.
* **Logger** : Gère l’écriture des actions dans le fichier log journalier.
* **StateManager** : Met à jour le fichier d’état en temps réel.
* **LanguageManager** : Gère la langue d’affichage.
* **Controller** : Interprète les actions de l’utilisateur et pilote le traitement.

Les relations entre ces classes garantissent la **cohérence de l’architecture MVC**, en séparant les responsabilités entre les données, les interactions et le contrôle du flux.



1. **Présentation de la logique du modèle MVC appliquée à EasySave 1.0 :**

**Présentation générale de l’architecture MVC**

Le modèle MVC, abréviation de Modèle - Vue - Contrôleur, est une architecture logicielle conçue pour séparer les différentes responsabilités d’une application en trois couches bien distinctes :

* **Le Modèle :** il représente la logique métier, les données et les règles de gestion. C’est la partie du logiciel qui contient les définitions des objets métiers, la structure des données manipulées, et les traitements appliqués à ces données sans aucune dépendance à l’interface utilisateur. Le modèle est indépendant des actions de l’utilisateur.
* **La Vue :** elle constitue la partie visible par l’utilisateur. C’est ce qui affiche les informations à l’utilisateur et recueille ses actions. La vue ne contient aucune logique métier, elle se limite à présenter les données issues du modèle et à relayer les actions vers le contrôleur.
* **Le Contrôleur :** il fait le lien entre la vue et le modèle. Il interprète les actions de l’utilisateur, déclenche les traitements nécessaires sur le modèle, et met à jour la vue en conséquence. Il assure la gestion du flux d’interaction entre les différentes couches.

Cette architecture garantit ainsi une séparation claire entre les règles métier, les interactions utilisateur, et les actions de contrôle, rendant le code plus structuré, plus modulaire et plus facile à maintenir et faire évoluer.

**Mise en œuvre du modèle MVC dans EasySave 1.0**

EasySave 1.0, qui est une application console de sauvegarde de fichiers, a été conçu selon ce principe de séparation des responsabilités.

**Le Modèle**

Dans EasySave, le modèle est constitué de toutes les classes qui portent la logique de gestion des sauvegardes et des fichiers. Ces classes assurent les principales fonctionnalités suivantes :

* La définition des travaux de sauvegarde : chaque travail de sauvegarde est défini par un nom, un chemin source, un chemin de destination et un type desauvegarde (complète ou différentielle). Ces informations sont encapsulées dans une structure dédiée qui permet de gérer plusieurs travaux en parallèle.
* **L’exécution des sauvegardes :** le modèle gère la copie des fichiers du répertoire source vers le répertoire de destination, en respectant le type de sauvegarde choisi par l’utilisateur.
* **La gestion des logs :** toutes les actions réalisées par l’application, comme le transfert des fichiers, la création des répertoires, ou les erreurs rencontrées, sont tracées dans un fichier log journalier. Ce fichier contient des informations détaillées telles que l’horodatage, les chemins des fichiers, la taille des fichiers et la durée de transfert.
* **La gestion du fichier d’état en temps réel :** le modèle met à jour en continu un fichier qui décrit l’état d’avancement des travaux en cours. Ce fichier contient notamment le nom du travail, l’état courant (actif ou non), le nombre de fichiers restants, la taille des fichiers à traiter et l’adresse du fichier en cours de traitement.

Toutes ces fonctionnalités sont regroupées dans des classes spécialisées et totalement indépendantes de l’affichage à l’utilisateur. Le modèle se concentre uniquement sur la gestion des données et des opérations métiers.

**La Vue**

La vue dans EasySave est représentée par l’interface console. Elle assure les rôles suivants :

* **Affichage des menus :** à chaque lancement, l’utilisateur est guidé par un menu textuel qui lui présente les différentes actions possibles comme créer un travail de sauvegarde, lister les travaux existants, exécuter une sauvegarde ou quitter l’application.
* **Saisie des informations :** la vue demande à l’utilisateur de renseigner les informations nécessaires pour configurer un travail, comme le nom du travail, le chemin source, le chemin de destination et le type de sauvegarde.
* **Affichage des résultats :** une fois une action exécutée, la vue affiche un retour clair, qu’il s’agisse d’une confirmation de succès, d’une liste des travaux enregistrés, ou d’un message d’erreur en cas de problème.

La vue se limite à ces interactions avec l’utilisateur. Elle ne contient ni logique métier, ni traitement de données. Elle est exclusivement chargée de présenter des informations et de capter les choix de l’utilisateur.

**Le Contrôleur**

Le contrôleur est le cœur de l’interaction entre la vue et le modèle. Il reçoit les actions de l’utilisateur via l’interface console, analyse les demandes, et pilote les traitements nécessaires. Le contrôleur dans EasySave prend en charge :

* **L’interprétation des commandes utilisateur :** par exemple, lorsqu’un utilisateur entre une commande pour exécuter une sauvegarde spécifique ou une série de sauvegardes, c’est le contrôleur qui analyse cette commande et lance l’exécution des travaux concernés.
* **La gestion des créations, modifications et suppressions de travaux :** le contrôleur vérifie les saisies de l’utilisateur, applique les règles de validation (par exemple, vérifier que le nombre maximal de travaux n’est pas dépassé), et transmet les informations au modèle pour traitement.
* **Le pilotage des sauvegardes :** une fois les travaux identifiés, le contrôleur orchestre l’appel des méthodes du modèle pour réaliser les copies de fichiers, mettre à jour les logs et enregistrer l’état d’avancement.
* **La gestion des langues :** le contrôleur permet également à l’utilisateur de basculer entre les langues disponibles (français et anglais) en adaptant dynamiquement l’affichage des menus et messages.

Grâce à cette gestion centralisée, le contrôleur garantit une exécution fluide et cohérente des actions, tout en respectant les contraintes définies dans le modèle.

Bénéfices de l’application du modèle MVC à EasySave

L’adoption du modèle MVC pour EasySave présente plusieurs avantages :

* **Séparation des responsabilités :** les rôles de gestion des données, d’interaction avec l’utilisateur, et de pilotage des actions sont clairement répartis.
* **Modularité** : chaque couche peut évoluer indépendamment des autres. Par exemple, une future version graphique pourra réutiliser les mêmes modèles et contrôleurs sans modifier leur logique.
* **Lisibilité et maintenabilité :** le code est plus simple à comprendre et à maintenir, car chaque composant a une responsabilité bien définie.
* **Réutilisabilité :** les composants comme la gestion des logs et de l’état peuvent être réutilisés dans d’autres projets, grâce à leur encapsulation dans des bibliothèques dynamiques indépendantes.

1. **Fonctionnement détaillé de l’utilisation d’EasySave 1.0**

EasySave 1.0 a été conçu pour être une application simple d’utilisation, tout en étant performante et flexible. Le parcours utilisateur est entièrement guidé depuis le lancement de l’application jusqu’à la finalisation des travaux de sauvegarde. Ce fonctionnement est organisé sous la forme d’un **menu principal** qui propose différentes actions accessibles en permanence.

**Déroulement général de l’utilisation**

Dès que l’utilisateur lance l’application, celui-ci est accueilli par un **menu principal** affiché dans la console. Ce menu propose plusieurs choix correspondant aux fonctionnalités principales du logiciel. L’utilisateur peut à tout moment :

* Créer un nouveau travail de sauvegarde.
* Modifier un travail existant.
* Supprimer un travail.
* Lister les travaux enregistrés.
* Exécuter un ou plusieurs travaux.
* Exécuter tous les travaux.
* Changer la langue d’affichage.
* Quitter l’application.

L’utilisateur navigue dans l’application en entrant les numéros ou les identifiants correspondant à l’action qu’il souhaite réaliser.

**Description détaillée de chaque action**

**1. Création d’un travail de sauvegarde**

Lorsque l’utilisateur choisit de **créer un nouveau travail**, le programme vérifie d’abord qu’il ne dépasse pas la limite maximale autorisée (jusqu’à 5 travaux). Si cette limite est atteinte, un message d’erreur s’affiche, informant l’utilisateur qu’il doit supprimer un travail existant avant d’en créer un nouveau.

Si la création est autorisée, l’utilisateur est invité à renseigner les informations suivantes :

* Le **nom du travail**.
* Le **chemin du répertoire source** à sauvegarder.
* Le **chemin du répertoire de destination** où les fichiers seront copiés.
* Le **type de sauvegarde** (complète ou différentielle).

Une fois toutes les informations saisies, l’application enregistre le travail et affiche un message de confirmation indiquant que la sauvegarde est prête à être exécutée.

**2. Modification d’un travail de sauvegarde**

L’utilisateur peut choisir de **modifier un travail existant**. Il doit pour cela sélectionner le travail à modifier en saisissant son identifiant. Le logiciel affiche alors les informations du travail sélectionné. L’utilisateur peut modifier les champs souhaités et valider ses changements. En cas de saisie incorrecte, un message d’erreur est affiché.

**3. Suppression d’un travail de sauvegarde**

La suppression d’un travail se fait en saisissant l’identifiant du travail à supprimer. L’application vérifie que le travail existe bien. Si c’est le cas, le travail est supprimé et un message de confirmation est affiché. Si l’identifiant est incorrect ou introuvable, l’utilisateur est informé que le travail n’a pas été supprimé.

**4. Liste des travaux de sauvegarde**

En choisissant l’option de **liste**, l’utilisateur peut consulter l’ensemble des travaux enregistrés. Pour chaque travail, l’application affiche les informations suivantes :

* Le nom du travail.
* Le chemin source.
* Le chemin de destination.
* Le type de sauvegarde.

Cette fonctionnalité permet à l’utilisateur de vérifier les travaux configurés avant de lancer une exécution.

**5. Exécution d’un ou plusieurs travaux**

L’utilisateur peut exécuter un ou plusieurs travaux en saisissant leur identifiant. Il peut saisir :

* Une **plage d’identifiants** sous la forme « 1-3 » pour exécuter les travaux 1, 2 et 3 dans l’ordre.
* Une **sélection multiple** sous la forme "1 ; 3" pour exécuter uniquement les travaux 1 et 3.

Le programme effectue alors la sauvegarde des fichiers en respectant les paramètres définis (chemin source, chemin destination, type de sauvegarde). Pendant l’exécution, le logiciel enregistre :

* Les actions effectuées dans un **fichier log**.
* L’état d’avancement en temps réel dans un **fichier d’état**.

À la fin du processus, l’utilisateur est informé de la réussite ou des éventuelles erreurs rencontrées.

**6. Exécution de tous les travaux**

L’utilisateur peut également choisir de **lancer tous les travaux** enregistrés en une seule commande. L’application enchaîne alors automatiquement l’exécution de chaque travail, en respectant les mêmes règles de suivi et de journalisation.

**7. Changement de langue**

EasySave propose deux langues : **français** et **anglais**. L’utilisateur peut, à tout moment, accéder à l’option de changement de langue. L’application propose alors le choix entre :

* « 1 » pour le français.
* « 2 » pour l’anglais.

Le changement de langue est appliqué immédiatement à tous les menus et messages affichés.

**8. Quitter l’application**

Enfin, l’utilisateur peut choisir de **quitter l’application** à tout moment. Le programme termine alors proprement son exécution et ferme la console.

**Résumé de l’expérience utilisateur**

Le fonctionnement de l’application est basé sur une **interaction continue et guidée**, où l’utilisateur est :

* Orienté à chaque étape par des **messages clairs**.
* Libre de naviguer entre les actions sans perdre ses travaux enregistrés.
* Informé en temps réel des **états d’avancement** et des **résultats** des sauvegardes.
* Assisté par des **mécanismes de validation** qui préviennent les erreurs de manipulation.

Cette expérience est renforcée par la **prise en charge multilingue**, la **gestion dynamique des travaux**, et la **traçabilité complète des actions**, offrant ainsi une solution fiable et professionnelle pour les opérations de sauvegarde.

**6)Conclusion :**

Le développement d'Sauvegarde facile 1.0 constitue une première étape réussie dans la mise en place d’une solution de sauvegarde fiable, accessible et adaptée aux besoins des utilisateurs. Ce projet a permis de concevoir un logiciel fonctionnel en ligne de commande, intégrant les fonctionnalités essentielles attendues : création et gestion de travaux de sauvegarde, exécution ciblée ou complète des sauvegardes, traçabilité via un fichier log, et suivi en temps réel de l’avancement des opérations.

La mise en œuvre du modèle Modèle - Vue - Contrôleur (MVC) a permis de structurer efficacement le logiciel, assurant une séparation claire entre la gestion des données, les interactions utilisateur et le pilotage des traitements. Cette organisation garantit une base solide pour les évolutions futures, notamment la transition vers une interface graphique plus conviviale prévue pour la version 2.0.

Les diagrammes UML réalisés ont permis de formaliser les cas d’utilisation, les interactions dynamiques, les flux d’activités, ainsi que la structure interne du logiciel. Ces éléments ont facilité la compréhension collective du système et la répartition des tâches au sein de l’équipe.

Enfin, l’utilisation d’outils de gestion de version comme Lien avec GitHub a permis de collaborer efficacement, de suivre l’avancement des développements, et d’assurer une gestion rigoureuse du code source.

EasySave 1.0 est ainsi une solution simple mais robuste qui répond aux besoins définis dans le cahier des charges, tout en offrant une architecture évolutive prête à accueillir les améliorations futures.