

|  |
| --- |
| Livrable 1 : EasySave V 0.1 |
|  |
| 11 FÉVRIER  Service : DSI ProSoft  Créé par : Hatice YILDIZ, Lisa JUNG, Aya SALEM, Ayse DENIZ |

Une image contenant capture d’écran, Police, Graphique, texte

Description générée automatiquement

Table des matières

[1. CONTEXTE : 3](#_Toc190174660)

[1.1 DSI ProSoft : 4](#_Toc190174661)

[2. INTRODUCTION : 5](#_Toc190174662)

[3. DIAGRAMMES : 6](#_Toc190174663)

[3.1 Diagramme Use Case : 6](#_Toc190174664)

[3.2 Diagramme d’activité : 8](#_Toc190174665)

[3.3 Diagramme de classes : 10](#_Toc190174666)

[3.4 Diagramme de séquence : 13](#_Toc190174667)

[4. Documentation de la Dynamic Link Library : 26](#_Toc190174668)

[5. Documentation Utilisateur : 26](#_Toc190174669)

[6. Documentation Technique : 26](#_Toc190174670)

[7. Release note V0.1 : 26](#_Toc190174671)

[8. Conclusion : 27](#_Toc190174672)

1. CONTEXTE :

Le projet EasySave, développé dans le cadre de l’éditeur de logiciels ProSoft, vise à concevoir une solution performante et évolutive pour la gestion et l’automatisation des sauvegardes de fichiers. Ce logiciel, structuré en plusieurs versions successives, repose sur une architecture modulaire favorisant la maintenance, l’évolutivité et la compatibilité avec les futures améliorations. En intégrant des fonctionnalités telles que la sauvegarde complète et différentielle, un suivi détaillé via des logs journaliers et une interface utilisateur optimisée, EasySave répond aux besoins des entreprises en matière de protection des données. Le développement suit les bonnes pratiques du génie logiciel, avec une gestion rigoureuse des versions, l’utilisation de C# et .NET 8.0, ainsi qu’une documentation garantissant une prise en main rapide et efficace par les utilisateurs et le support technique.

## DSI ProSoft :

Nous sommes une équipe de 4 membres, représentant la DSI ProSoft, qui vise à concevoir une solution performante et évolutive pour la gestion et l’automatisation des sauvegardes de fichiers autour d’un nouveau logiciel qui sera implémenté « EasySave ».

Lisa JUNG

Une image contenant personne, Visage humain, sourire, mur

Description générée automatiquement

Ayse DENIZ à Chef de projet

Une image contenant personne, Visage humain, sourire, habits

Description générée automatiquement

Hatice YILDIZ

Une image contenant Visage humain, personne, habits, sourire

Description générée automatiquement

Aya SALEM



1. INTRODUCTION :

Dans le cadre du développement du logiciel EasySave, ce premier livrable correspond à la version 1.0 du projet. L'objectif principal est de concevoir une application console sous .NET Core, permettant de gérer et d'exécuter des travaux de sauvegarde de manière efficace et modulaire.

Cette première version devra offrir les fonctionnalités essentielles suivantes :

* La création et la gestion de jusqu'à cinq travaux de sauvegarde, définis par un nom, un répertoire source, un répertoire cible et un type (sauvegarde complète ou différentielle).
* L’exécution de ces sauvegardes, soit individuellement, soit en séquence, avec prise en charge des disques locaux, externes et lecteurs réseau.
* La journalisation en temps réel des actions effectuées via un fichier log journalier, contenant des informations détaillées sur chaque transfert (horodatage, chemin des fichiers, taille, temps de transfert).
* L’enregistrement en temps réel de l’état d’avancement des sauvegardes dans un fichier JSON unique, permettant de suivre l’évolution de chaque tâche en cours.

Un aspect clé de ce livrable est l’intégration d’une Dynamic Link Library (DLL) pour la gestion des logs. Cette librairie devra être réutilisable et garantir la compatibilité avec les futures versions du logiciel. De plus, le projet devra être conforme aux bonnes pratiques de développement exigées par ProSoft, incluant une architecture modulaire, une gestion efficace des fichiers JSON, ainsi qu'une compatibilité avec les utilisateurs francophones et anglophones.

Enfin, si cette version répond aux attentes, une version 2.0 avec une interface graphique (WPF) sera envisagée pour améliorer l'expérience utilisateur et enrichir les fonctionnalités du logiciel.

1. DIAGRAMMES :

## Diagramme Use Case :

Un diagramme de cas d'utilisation (Use Case Diagram) est une représentation graphique des interactions entre les utilisateurs (acteurs) et un système. Il illustre les fonctionnalités principales offertes par le système sous forme de cas d'utilisation, permettant ainsi de comprendre qui utilise le système et comment.

Ce type de diagramme est essentiel en phase de conception, car il permet de clarifier les besoins et de définir les interactions entre le logiciel et ses utilisateurs sans entrer dans les détails techniques.

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

Le contenu généré par l’IA peut être incorrect.

Explication :

Ce diagramme de cas d'utilisation illustre les principales fonctionnalités du logiciel EasySave et les interactions possibles entre l'utilisateur et le système. Elle est composée de :

Acteur :

Utilisateur : L’acteur représenté par un bonhomme est celui qui interagit avec le logiciel pour effectuer différentes opérations de sauvegarde.

Cas d’utilisation principaux :

* Réaliser une sauvegarde

Fonctionnalité centrale du logiciel, permettant à l’utilisateur de gérer ses sauvegardes.

* Créer un travail de sauvegarde

Avant d’exécuter une sauvegarde, l’utilisateur doit créer un travail de sauvegarde, définissant :

* Un nom
* Un répertoire source
* Un répertoire cible
* Un type de sauvegarde (Complète ou Différentielle)
* Exécuter une ou plusieurs sauvegardes

Une fois le travail de sauvegarde configuré, l’utilisateur peut lancer manuellement une ou plusieurs sauvegardes.

Il peut choisir de lancer toutes les sauvegardes en séquence ou de sélectionner des sauvegardes spécifiques.

* Consulter l’état temps réel des sauvegardes

Permet à l’utilisateur de suivre en direct la progression des sauvegardes.

Les informations incluent :

* Nom de la sauvegarde
* Source
* Destination
* Le nombre total de fichiers
* La taille totale des données
* Le nombre de fichiers restants
* Le progrès de la sauvegarde
* Consulter le fichier log journalier

L’utilisateur peut consulter les logs journaliers générés par le logiciel.

Ces fichiers enregistrent toutes les actions de sauvegarde et leurs détails (date, heure, état de la sauvegarde, erreurs éventuelles, temps, source, destination).

## Diagramme d’activité :

Un diagramme d'activité est une représentation graphique des flux de contrôle et des processus d'un système. Il illustre la succession des actions et des décisions effectuées au sein du logiciel, souvent sous la forme d'un workflow.

Ce type de diagramme est utilisé pour modéliser le comportement dynamique d’un système en mettant en évidence les différentes étapes d'un processus, les choix possibles, ainsi que les conditions de transition entre les actions. Il est particulièrement utile pour visualiser les algorithmes, les flux de travail métier, ou encore les enchaînements logiques d'exécution.

Une image contenant texte, diagramme, croquis, ligne

Le contenu généré par l’IA peut être incorrect.

Lien vers le [diagramme d’activité](https://mermaid.live/edit#pako:eNqVVttu2jAYfhXLN20l6IGWHrjYVLXdTbtVU6ddDCrLSwxYSmzqw1ZGeZdejufIi-23E4gDSemQEOT3933_2coMRzJmuIeHifwdjaky6Nv1QCD4XGeLn9bs5j97qN3-gO6oiFjKhCF0Mkl4RA2Xou-tCiU7gfEx16gleKmrseTPJKFiZFnfPWgOEhTlloIegjyLDoc8GjNFQNOuaDQKXFYgnvTVcmOY6he_tYEuzxz-Exe78N1r0rtS2cJziab2FxtRFUMKYAS-FQyV1saQbp6zRWTXNQprLnMQMx1I6UatKym0TUweUSJHul9YXKqg4UwF-YG6ehFQM1ysp7Bqi4OY6YRV0itKbQVyRyiuSbOuLl60cCtkGkrmVqcIB7WCtbxQUUurIlYrCnL-8C3hDXqoXV-k0EGAqPVyaUkqudDkKBAgTLAR18aNi3f3cm_5SzHpQbtJ9krYch6CBTlIqmOBsle0wgVrU9NE7-57tlB8WAw_mVBF0-yvUUzP7kAseIb6CdPmiUaRFEOpUjB9nOcOGkWCjD4zremIkahmKsj9bf-yGGS0AlSLiO5vH9_l7YsUpbfylKRUPFkqTOnJCr9UoQK08MAliriIpFIsMqtN25LA5n1UpdVE0nh75MzaS8Fztk7S7EabNgz7k92hFnksOgqLCXkiJlQ5eq6V_zWmlTJTG8EGVPGluC5LngPDSMTOeiyVy75pBfI6eFkXAIeR1MSwdKKJ00hKl25BuJ9YX2boMdxZDog8cLmczVLbOrsl-Sb6xj293lrH4EBxZ5A46DX0tQC6ffE6vqAO72ZsuaNvCzf1tIIta8oT6NsU0byf6wE8vt-jvxqWsg3OEtbgoDnSpprXetoE4xZOGcwLj-FNaOaoA2zG8OIywD34G7Mhhc4N8EDMAUqtkQ9TEeGeUZa1sJJ2NMa9IU00PNlJTA275nQE-7-yTqj4IWXlGfdm-Bn3Ohfd_e555_T04rxzcnh2fNzCU9w7nLfwH0842j_MP52T7ln34vS8hVnMjVSf8xc3__42_wco5aJi).

Explication :

Au lancement de l’application EasySave, un menu interactif permet à l’utilisateur de choisir une langue ou de quitter l’application.

Dans le cas où l’utilisateur a choisi une langue, un autre menu lui permet de choisir une action parmi la création d’une sauvegarde, l’exécution d’une ou plusieurs sauvegardes enregistrées, ou encore la consultation du fichier de log journalier. Depuis ce deuxième menu interactif, l’utilisateur a également la possibilité de quitter l’application.

Si l’utilisateur choisit de créer une nouvelle sauvegarde, une vérification du nombre de sauvegardes enregistrées est effectuée. S’il existe déjà 5 sauvegardes, un message en informe l’utilisateur. Sinon, l’utilisateur est invité à entrer les différents paramètres d’une sauvegarde à savoir, un nom, une source, une destination, ainsi qu’un type. Lorsque tous les paramètres ont été saisis, une vérification permet de savoir s’ils sont conformes au format attendu. Si un ou plusieurs des paramètres ne sont pas conformes, un message le fait savoir à l’utilisateur. Dans le cas contraire une nouvelle sauvegarde est créée et enregistrée, puis un message en informe également l’utilisateur.

Si l’utilisateur choisit d’exécuter une ou plusieurs sauvegardes, une vérification permet de contrôler l’existence d’au moins une sauvegarde. Dans le cas où aucune sauvegarde n’a été préalablement enregistrée, un message en informe l’utilisateur et celui-ci est redirigé vers la création d’une sauvegarde. Sinon, l’utilisateur est invité à saisir la ou les sauvegardes qu’il souhaite exécuter. Les sauvegardes renseignées sont alors exécutées et leur état en temps réel est affiché.

Si l’utilisateur choisit de consulter le fichier de logs, une vérification permet de contrôler l’existence d’un fichier de logs du jour. Si ce n’est pas le cas, cela signifie qu’aucune sauvegarde n’a été lancé le jour même. L’utilisateur est alors redirigé vers l’exécution d’une ou plusieurs sauvegardes. Dans le cas où un fichier de log est présent, son contenu est affiché dans la console.

Lorsque l’action choisie préalablement par l’utilisateur est terminée, le menu permettant le choix d’une action est à nouveau affiché.

Enfin, si l’utilisateur choisit de quitter l’application dans l’un des menus, le programme se termine et l’application se ferme.

## Diagramme de classes :

Un diagramme de classe est une représentation graphique des structures statiques d’un système logiciel. Il décrit les classes, leurs attributs, leurs méthodes, ainsi que les relations qu’elles entretiennent entre elles (association, héritage, agrégation, composition, etc.).

Ce type de diagramme est essentiel en programmation orientée objet car il permet de visualiser l’architecture du code avant son implémentation. Il aide à mieux structurer le projet, favorise la réutilisabilité et assure une bonne compréhension du modèle de données du logiciel.

Une image contenant texte, diagramme, Plan, ligne

Le contenu généré par l’IA peut être incorrect.

Lien vers le diagramme de classes : [Online FlowChart & Diagrams Editor - Mermaid Live Editor](https://mermaid.live/edit#pako:eNq1WVtT2zgU_isebx8MJEyhhEKG6QyFXuiEtrOh3ZlddhjFFkbFkVLJZklZ-O0ryZYt6-IEwubFifSdiz6dc3Ss3IUxSWA4DOMMMHaMQErB9BwH_IPBFLIZiGEwBsUNTAFN4F05JT5SIHgL4utiNs4pyGE6fw_inNC5hhKfjeCIQj7fxkZrhvB6I3V_jk1DR2Q6y6BSspKhTjvH6PISUohzBLL_21Ybahl4dwvjQlmIWE4RTntB-VwLbghK1heoZ7DTxG_BiKQppMGw-mJOT6T0dwT_4ZC39Y-VHBWfPqeZQknru1vE8kjHmtqrXUAZNFS3TTwielZg_JGx8yyWKsa4oi-XI8Byp1AjLIBnaApbrrY8bjL7jIKkiHNEsJXZJ3wOs4wr8y7iDNAU5iOA0wKkkMd-6UJ3VL7nBMVXy-rqVPUOpxliT9OlPR5d7o4IzinJMkjvzF0qASMe0yKp-OOhHDnlVTZ7MNEXzvJZ55oxbkmvkJ7uVNNLGqcNsDmOOZvs2gR-BZTBT2RyghN4W6uU60U4f1gx9NTOdfA8FXwKiiuo5NcE3ZTMKIyLG9tU1FLZa0m7WfsMblDKeVNIFrkz-HA2y-YKNIYZlKvn4Akh2TKUHUoBZvF1CnHh5Wqj5krAHDxtKJ7EvIsjplyVu82BfItNTNuFqDbVq7W6uTsFmJNRLawj5PoqiktoxF3QwQ-CQiu9PgKcZFU4y9Au6fbb0AVKe36JJ0W2qqntuSqdEWY5wFzBUMdZsIzE1xzC15MQnM0DMvnBt-eFyewFq6oHx9q13ESfVLb5YjXbltIxzGu9ka3Vvcc1rC4U2saVQ0tVjIV1WUbcnZ2eU0GpeQZUCyIFjf3T5Unin57P_LKaT56z-hGthToQOItPWaDonr6C_GrBQhfCpAN-JSQHmdDBzsgRmc3dpaJBjdEvoSwjtqrf4RQgzG1IoFvRV0pSCpk1203fJ04IBpmVfkxMagTU-VWu9YWJF2n4ZfJjYSb2-TCD9AZSplqCky_VyIOVYIdJoiajGlbGh6X3M8nR5VyBWKQt0imxIdvgQqRrk4x2LGpqLAXfZglQ8ovNlWi1Ta3Ow1rMiICk1LsW6I2THHuw4GNwU7lhYS0D_orykcsSin4WdkX5kJEJyL4DisCER6D15pR1vTltiNGOZFqpK3JXubr5UEFmFVZVwcVhDhOFdyTPE7oPt0-VxAKPLsymum5mm6FnOnad3HlPXvtIfeajd4WjTnRUFt_feCU4ZNethsXz_tF69dHfkzEjGXRrl9lsp_0zhbXT5jFiswzMm85aD6ReILvAZX3oit8u46o5dZj2vyBVsieY11teAKez7juOowwCOo4phDhafkUdUVKfHUtv47rfYqlS1nEw1RSK9h3hamF__R3wHoLppLxYovBeHM_5MIpHCF-P0ISC1k1bFSKyxN75T2DPwZu1irBxoD-uYtcY7kjk3sMPMK90iIasvnWwqsMfFPF3RZJG4_LgbT9kN9QLElLwY6cnXw0XBcTBAQ9GSC85sW_edF0ddAEVAkxE8YpzCTDvD11qfAXOQKl4VE67bz42N91-OW9_vWjvHa5Xwu3Nwb_9vtv0QimvCy7JUsRHtz3ucmw5Gf8NpVduvd9v9TgeiHn95IDp5DeNuGuDlxLxr6VTrLx_LOnwxW51sdgNstqWhgbtlqUD5czRDrxOrgHqWrATbF0w2BdhARE7b9-vLUA2XjqATtPtuyOpzLiq8iAaUwZAmDEaeC_OckcbkOR3AsjCANE3pIQ3pdCDMnazxBnrsfKypkyLGLOBbvfB0hvv6mCjrHuB8v55cQAaMCtF6nnLo5owVffqHlXVjnrATZ0OcKWTiqVF5p3Kq7ZITjqyxTXdWHbNmpumY4y8MKfcittBH_bCKaRTgJJwGMqe6jzMr-AUnodD_jUB9Po8PMf3HAeKnIx5MxcOc1rAXljI1rH6k1gNwgTx6nVa_YssHr1wBnA4vAtvw2F_d-vV5u7uYLCzszXYH2xv7_TCOR_eGexv7u5tv97d2tre2t8bvLrvhb8I4Vp3Nrdfvtx7vT14Ndjd23u9t98LKSn)

Différents types de relations entre deux classes ont été utilisées dans le diagramme de classes :

*  : une flèche vide avec un trait en pointillés représente une dépendance. C’est-à-dire que la classe d’où part la flèche utilise la classe vers laquelle la flèche est dirigée.
*  : une flèche vide avec un trait plein représente un héritage. Cela signifie que la classe vers laquelle est dirigée la flèche est une fille de la classe d’où la flèche part. Dans ce type de relation la classe fille est un type plus spécifique de sa classe mère et implémente les mêmes attributs et méthodes que celle-ci.
*  : un losange vide avec un trait plein représente une agrégation. Cela veut dire que la classe vers laquelle le losange est dirigé contient un ou plusieurs objets de la classe d’où part le losange. Mais les instances de la classe d’où part le losange peuvent vivre une fois l’autre classe détruite.
*  : un losange plein avec un trait plein représente une composition. Cela veut dire que la classe vers laquelle le losange est dirigé contient un ou plusieurs objets de la classe d’où part le losange. Mais les instances de la classe d’où part le losange ne peuvent pas vivre une fois l’autre classe détruite.

Explication :

Dans le diagramme de classes de l’application EasySave nous trouvons plusieurs patrons de conception.

En premier lieu, nous avons un patron de conception nommé usine. Il permet de centraliser la création d’objet dans le but de découpler l’instanciation de classes spécifiques du reste du code.

Dans notre cas, le patron de conception usine est formé par l’interface BackupStrategy et ses deux classes héritières CompleteBackupStrategy et DifferentialBackupStrategy qui permettent d’exécuter une sauvegarde en fonction de son type (complète ou différentielle). De plus, il y a l’interface BackupStrategyFactory et ses deux classes filles CompleteBackupFactory et DifferentialBackupFactory dont l’objectif est de créer des objets de type CompleteBackupStrategy ou DifferentialBackupStrategy.

L’implémentation du patron de conception usine respecte le principe SOLID ouvert/fermé. En effet, si nous devons ajouter un type de sauvegarde il nous faudra uniquement ajouter une classe héritière de BackupStrategy et une classe fille de BackupStrategyFactory. Il n’y aura alors pas nécessité de modifier tout le code.

En second lieu, nous avons aussi un patron de conception appelé stratégie. Il permet de changer dynamiquement le comportement d’un objet en déléguant une partie de son comportement à des objets appelés stratégies.

Dans notre cas, le patron de conception Stratégie est formé par l’interface ITranslateStrategy ainsi que ses deux classes héritières French et English. Ces dernières permettent de déterminer la langue dans laquelle l’application est traduite (Français ou Anglais). Il y a encore la classe Translation qui implémente une instance de type ITranslateStrategy. Elle permet de traduire du texte dans la langue déterminée par ITranslateStrategy.

L’implémentation du patron de conception stratégie respecte, lui aussi, le principe SOLID ouvert/fermé. Effectivement, si nous devons ajouter une langue pour l’application il nous faudra uniquement ajouter une classe héritière de ITranslateStrategy. Il n’y aura alors pas nécessité de modifier tout le code.

En troisième lieu, nous avons un patron de conception singleton. Il garantit qu’une classe ne peut avoir qu’une seule instance et fournit un point d’accès global à l’instance en question.

Dans notre cas, le patron de conception singleton est mis en place dans la classe Translation. Pour ce faire, le constructeur de la classe est redéfini. Plutôt que de créer une nouvelle instance de la classe Translation son constructeur vérifie tout d’abord s’il existe déjà un objet de cette classe. Si tel est le cas l’objet existant est retourné, sinon une nouvelle instance est créée. De cette manière, il y aura toujours une seule et unique instance de la classe Translation dans le code.

L’implémentation du patron de conception singleton limite le nombre d’instance d’une classe à 1. Cela peut être utile dans certains cas, notamment lorsque l’instanciation d’une classe est couteuse en mémoire, il est alors préférable de garantir l’unicité de son instance.

En dernier lieu, nous avons encore un patron de conception nommé observateur. Il permet à un objet (le diffuseur) de notifier automatiquement plusieurs autres objets (les souscripteurs) lorsqu’un événement se produit.

Dans notre cas, le patron de conception observateur est formé par la classe BackupStateJournal qui a le rôle de diffuseur. Ce dernier est également relié à l’interface IObserver ainsi que sa classe héritière ConsoleView. Cette classe implémente une méthode qui retourne les informations comprises dans les notifications de la classe BackupStateJournal. En outre, le diffuseur implémente plusieurs méthodes qui ont pour objectif, entre autres, d’ajouter un objet à la liste des souscripteurs ou encore de notifier tous les souscripteurs.

L’implémentation du patron de conception observateur permet d’ajouter facilement des souscripteurs sans modifier tout le code. Effectivement, il nous suffit d’appeler la méthode adéquate du diffuseur. Un nouveau souscripteur sera alors ajouté à la liste de diffusion et notifié comme tous les autres souscripteurs. Le patron de conception observateur permet également d’automatiser les réactions dans le code sans devoir le modifier.

Enfin, l’utilisation de ces différents patrons de conception dans le code a encore pour objectif la réutilisabilité des différentes parties du code sans devoir le modifier.

* 1. Diagramme de séquence :

Le diagramme de séquence UML est un outil graphique permettant de représenter l’interaction entre plusieurs composants d’un système au fil du temps. Il est utilisé principalement en modélisation logicielle pour visualiser le flux d’exécution d’un programme.

Dans le projet EasySave, le diagramme de séquence est essentiel pour représenter le fonctionnement du processus de sauvegarde.

Un diagramme de séquence fait partie des diagrammes UML. Il permet de :

* Décrire l’ordre chronologique des interactions entre les composants.
* Visualiser la communication entre objets et acteurs d’un système.
* Aider à comprendre le flux d'exécution d'une fonctionnalité.
* Servir de référence pour le développement et la documentation technique.

Composé essentiellement de :

* Acteurs :

Représentés par un bonhomme   
Ils symbolisent un utilisateur ou un système externe qui interagit avec l’application.   
*Exemple concret :* Un utilisateur qui déclenche une sauvegarde.

* Classes/Objets et participants :

 Représentés par des rectangles en haut du diagramme   
Ils correspondent aux différents composants logiciels qui participent à l’exécution du scénario.

*Exemple concret :* Une chaîne de restaurants :

* L’acteur serait le client qui passe commande.
* Les classes/objets seraient les différents services du restaurant :
* Serveur (BackupController) qui prend la commande et la transmet à la cuisine.
* Chef cuisinier (LogController) qui prépare la commande et note les plats réalisés.
* Responsable de salle (MenuController) qui organise le bon déroulement du service.

* Messages et interactions :

Flèches dirigées entre les objets   
Elles indiquent les appels de méthodes et les retours de valeurs.

Il existe différents types de flèches :

*  : ce type de flèches représente des messages synchrones dont l’expéditeur attend une réponse de la part du destinataire avant de pouvoir continuer son exécution.
*  : ce type de flèches représente des messages asynchrones dont l’expéditeur n’attend pas de réponse de la part du destinataire avant de pouvoir continuer son exécution.
*  : ce type de flèches représente des messages asynchrones qui sont utilisés lorsque le destinataire d’un message synchrone répond à son expéditeur.
* Une image contenant blanc, Police, clé, ligne

  Le contenu généré par l’IA peut être incorrect. : ce type de flèches représente des messages asynchrones qui sont utilisés lorsqu’un nouvel objet d’une classe est créé.
* Une image contenant symbole, ligne, Symétrie, conception

  Le contenu généré par l’IA peut être incorrect. : un message avec une croix au bout de la flèche est utilisé lorsqu’un objet d’une classe est détruit.
* Barres d'activation :

Représentées par des rectangles verticaux sur les objets   
Elles indiquent qu’un objet est actif et en train d’exécuter une tâche.

* Boucles et conditions :

 Représentées par des blocs alt, loop, opt en UML   
Elles permettent d’ajouter des conditions et des répétitions.

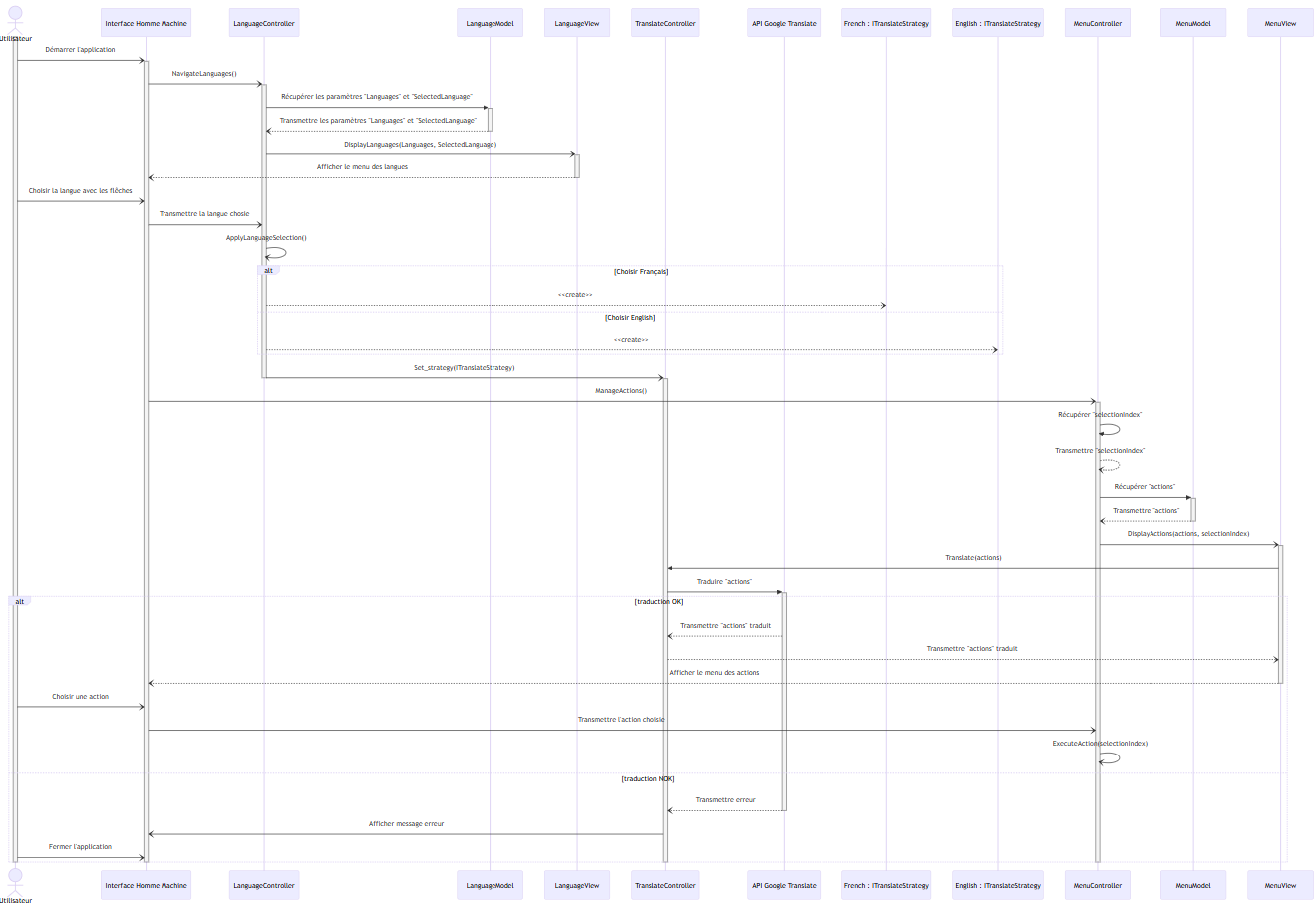
Dans notre projet, nous nous sommes appuyés sur le diagramme de cas d'utilisation (Use Case Diagram). En effet, chaque bulle dans ce type de diagramme représente une action, ce qui nous a permis de structurer notre approche.

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

Le contenu généré par l’IA peut être incorrect.

Ainsi, nous avons conçu cinq diagrammes de séquence, chacun illustrant un processus spécifique. Voici ces diagrammes accompagnés de leurs explications détaillées :

1. Diagramme de séquence pour le choix de la langue et de l’action :



Lien vers le [diagramme de séquence pour le choix de la langue et de l’action](https://mermaid.live/edit#pako:eNqtVltv2jAU_iuWX9ZqUBVouUQVEuplRRvdtK57mJAmKzkES7kwx-nKqv6erfsb_LEdO45JSEL7MB7A2N_5fPydi_1I3dgD6tAEfqQQuXDBmS9YOI8IfpgrY0HuEhCEJeRO8oAnTEIqsuUVE5K7fMUiSabXM4WZRhLEgrlAruMwBDJj7pJHUMV_YJGfMh_O40iKOAhANGNm6GLQvPyVw8_q6hfBoiRAb_ftMPk0VV6rn3dx7AewNauCrwQKtFR4M3LI1MJvpcBvf101u4x8lE3b5cPXGc4gSvc5r9YbpFFLmSw2kPwet9GxzKZ0VNuH5C1GziEXm-eQCYFTwRu2WgXcZZLHxlrFViGrMXPIDbvnPjLna8nBYWZUBZP2eLwl0Z475PPm2U1Xm2e9NSTqFCzc_JECx3NqWeeUgMSJWwjAleDlC3Na3k2zkjZ6267zVqsegkT6_7Fb8WwFfZTyKClPVgFbb4WxoxbZJd7RTBFkh9DBmSwW3F1qgUiIkSUeuhsoLLpaCqaGny9jnnBEMwMi7B5cfd5FsPmLTEkpri8KZXncZZxw2KdBHdcE88nKkJ0cUyvPExZI6_EV7rr5zbjxr2kb3CerQIecnbkCMP3G48wEggQsnSm3F8kMroYt8vYGvKbHOBhb-T0xRX1QrXNzbA9sTTa1wrzsyn3AwZ4aIXqiVbTlVgbpUtu1K5baHBu-CcQ08uAhz-xdGtx_l6aYGq-lUYVv29WuJyw7SdG2UMb7t6-x3YlR3gptQebCGdMWKR-hoGdWhuh7bZztZM5kLGvAmQB4y2grL-V1vqtCkGpVT5OP77d5q-4nJUazH1VBMi4utyy1jpkAZwq9mmqrzv4mlVNYw9pGlUZgkFugSf590cebKpPK1TxQdq-cBrs8lw_gptIU0UFtBuhWUgjITV1E2i-GBPBWzR9MjVEwglgNQ0gSrPGKcUm-KxBh_YVt-1ahyzS-hwqYuhdHYXn7eChMoiu0RdGRkHEPH5KPCjGncgkh3pcODj1YsDSQKs-fEMpSGd-uI5c6UqTQoiJO_SV1Fgz1btF05SGreYXmEHzTfIvj0ILwP3Ue6QN12p2jk06vP-ifHg_6_eHpqHfaomvq9EZH3eN-tz_qdfoD_AyfWvSXpugeDY4Hw97JybDbGXVHo97TP_rmxMc).

Explication :

Ce diagramme de séquence illustre le processus par lequel l’utilisateur peut choisir la langue d’affichage du logiciel, ainsi qu’une action à réaliser.

Au démarrage de l’application, un menu interactif est affiché sur l’Interface Homme Machine pour permettre à l’utilisateur de choisir la langue dans laquelle il souhaite continuer sa navigation.

Une fois que l’utilisateur a choisi sa langue préférée, un objet de la classe French ou de la classe English est créé. Celui-ci servira pour les différentes traductions nécessaires au sein de l’application.

Ensuite, un autre menu interactif est affiché sur l’Interface Homme Machine afin que l’utilisateur puisse choisir une action à effectuer. Bien sûre, avant d’être affichée, chaque option du menu est traduite dans la langue définie plus tôt. Mais si une erreur survient lors de la tentative de traduction, un message d’erreur s’affiche à la place du menu interactif.

1. Diagramme de séquence pour la création d’une sauvegarde :

Une image contenant texte, diagramme, Parallèle, Rectangle

Le contenu généré par l’IA peut être incorrect.

Lien vers le [diagramme de séquence pour la création d’une sauvegarde](https://mermaid.live/edit#pako:eNq1mMFy2jAQhl9Fo1OYAAOEQuJDZijpNDkw05k0PXS4aOwFlNgSleQ0aSbv0mN5Dl6sK-MEgw3YxvElwdr9Vvr1W2Z5oa70gDpUw68QhAtXnE0VC8aC4MVcIxW506AI0-TOcJ9rZiBUq-E5U4a7fM6EITfXIxtzIwyoCXOBXMsgADJi7owLSMePQIRDKYySvg8ZvM_MfQjnhyNGOH1_1-APDr_TY98VE9rHdeyjD77d2PXYP1-lnPqwThuLd3X4I36OBNq6tbNGJGajRk5RMIcMZ5JrrsiYDtVygSOhAKJZ-AhTpjwY01WSFdfmbIrmrKoEYIyCqLQUxI2IseCb8RaxTbhmwvMhFlsBs4yT2q7s0-1dwRXYpBjwlujBuw5Z28x8Qz4lVqkJBhBvubhf_iXLhVkuiItqoB56lWCv7cqkcXlJTjNkdtban4zpZh2dKARCwZRrY3XXzTGtrWtlUFfl0A0R3gs5Kl6EvoZbRzVQy8buucc7WoRPTDQrc2ARtm56D6srmt6lGomMPphMuDsDdQQffA0kkFxoNFgVBnr3sz0mnOjJHOiHk9p2jh3OabcveBv-EDwthAzsNH2WmCdxytnsMLWcvQ5zC9tqJeXxZZLKb1iITcsi307eCHbLonMXs9cB8Sm7ayEbsaWdgfOUocL3Y9Xm2As-zh970ZVapGClnC4pSM0yygqQ2yvb4UfYBY83w0X0Yv4AzxyiH2ucQ_yK3VO8XG4LFUdn-ShByW2mzJzCjhrKYO4v_-HX11KWSaSX80QCUMmm7-UVVueKTybLhQJhOOBoOY1SkHJKpTCV6JWDWtxTUbOh9epVbJ7n1b3WcqJLejEfvBqflq114GT6GK5tH58imt5xkL21rDYm9xG2Gbw5g9NEI5_6xlWPX6X15ClY3-Il-s3tvn9dLuKv5xiXy267d6Vm9E6DexmaWHkhkYC3k1uAHQmO-NjVwNHN7O0ay2xZnHrJxjWbVO5hymZV2JDmLZDV4TVSz00eGggv9UsGglL37INB6zQAFTDuUYe-2JAxNTMI8IR18F8PJiz0jRX3FUNZaOTts3CpY1QIdapkOJ1RZ8Kwua3TcO4hNv4J7v3unImfUm58ps4LfaJOp91tnnUvLs67_X6r1el3O3X6TJ12s9_pts465xe9frfV7_XOXuv0T4RoN1vx1em1L1rds9f_5TTHlQ).

Explication :

Ce diagramme de séquence illustre le processus par lequel une sauvegarde peut être créée et enregistrée dans le logiciel.

Lorsque l’utilisateur choisit l’action de créer une sauvegarde, plusieurs scénarios sont possibles.

Tout d’abord, l’application doit permettre d’enregistrer au maximum 5 sauvegardes. Une vérification est alors effectuée pour contrôler le nombre de sauvegardes déjà en mémoire.

Dans le cas où 5 sauvegardes sont déjà enregistrées, un message pour en informer l’utilisateur est traduit puis afficher sur l’Interface Homme Machine.

Dans le cas contraire, si moins de 5 sauvegardes sont enregistrées, l’utilisateur peut créer une nouvelle sauvegarde.

Nous savons qu’une sauvegarde est définie par un nom, une source, une destination, ainsi qu’un type. Un message va alors être traduit pour demander à l’utilisateur de saisir le nom de la sauvegarde. Un autre message affiché sur l’Interface Homme Machine, préalablement traduit, demandera à l’utilisateur d’entrer la source de la nouvelle sauvegarde. Un dernier message, traduit puis affiché sur l’interface, permettra à l’utilisateur de saisir la destination de la sauvegarde à créer. Enfin, un menu interactif, traduit avant d’être affiché sur l’Interface Homme Machine, permettra à l’utilisateur de choisir le type de sauvegarde qu’il souhaite créer parmi une sauvegarde complète ou différentielle.

Ensuite, une sauvegarde sera créée avec les différents paramètres entrés par l’utilisateur. Cette nouvelle sauvegarde sera également ajoutée à la liste des sauvegardes enregistrées dans le logiciel.

Pour finir, un message informera l’utilisateur de la création de la nouvelle sauvegarde.

1. Diagramme de séquence pour l’exécution des sauvegardes :

Une image contenant texte, diagramme, Parallèle, Plan

Le contenu généré par l’IA peut être incorrect.

Lien vers le diagramme de séquence pour l’exécution des sauvegardes :

[Online FlowChart & Diagrams Editor - Mermaid Live Editor](https://mermaid.live/edit#pako:eNrFWEtuGzkQvQrRG0uA5UEQZKNFgEzi_GA7RjzJAANtqO6SzIQiO_w4FoIcILeYZXSOvtgU2R_1h2wpxgzGC0sii8Wq4qtXRX5LUplBMk80fLEgUnjB6FrRzUIQ_KOpkYp80KAI1eSDYZxpasCqcjqnyrCU5VQY8ub1pZPBj-HcJQj7XAqjJOelqu5IeMVHBl9rWfd9KPU7TT_bvKu5PxZbdWMUerLevvQ-bvdLexOh9Rq6ouXi_uhw5XO5yTmYwOrwzFDDC7ZagQJhGOVDLfHZoaYLue4GrjMQjxqqeyutEpS3Q7YfDSDj3RIBdFfu0vwIRUdoyaE-9tbPoewrLpeUf6SK0SUH7eR7Q0GP142r7tvkxcXFdCFKSQ_y2dOnHshz8vxWMs0UWSR_KCo0RwfJ-T2k1qCYpvYO1lRlMNHTRdLkCrtzYk5Tb6jJCqd8Nu3Bf04uqaBreIbi6PRkOiKLplUmbcAYBX4PKUjq7YWBMaFEG53sbRcw9jUVGa-QWoYEDaiNHi7vJ-QcYapzTrfn95ppw8S6lGj8bswLp_IxO7wCUw9aYWrFGYx6PmCTWcD5Nh4u0HxArboFCE0mT8iG3jewaG16nD-4KZ682-nZasXSWwTGYI8MIygF8zhPiLS_HVK1N_qZTa2AljaC4vau2MEZeQ8Gk5hQSzao6ax2IQwJn6fNYb4RuTXXCmnMDM6xS-DN2so8j-hz4fLKCiLsptgpiR6TR6T4mzwhuTMJ6tRDEWlJzi2C3apOVE5PHs0en_TkM4t60KHH-M_IDTUMyxx6ZwLnM7TzEM6uqdJIfMs3IoP7CXMxOA7ER2LtZn9Kuthx8Lku8Kwejq6S4XrKAQkEw6ZjJ46Lhs5XdFhOTB7qdyisWE-UuYKvkymZpLdKCkk4FWmxm8a0NAZ2CtTc11Vr_GDEwmHtCiia7QvD-2KXgzKSIfeCZzDAWDrzKlg5fIJS4BKpSeBFolFVCkScwL1fk1MdPcSYTRGPO_0K-qyA9juJqPOBXifcH83CIEAYOcFit2aOptq8kioEKo7llmnCkWe2OVR1asz1gEWUG9JCq2-Wip8GytkYrsI91TwyXkeoE6WxtiwOZz3Yspsq5cdLxhGToV1jreTwcDxtBA-ldQx1tAhgd5ga26KPwRHEN26JjUUFuIb2WbmWFAnd9aSAxo2FDvEVb2DnI3PBozvUC7u_fhfpTCh7Q99C4Fd3SFd0A8EtStH9RNVWRjPlNSa-VK4A-YLO5VoTzJHU5hihYhc5lP4uvx44_Hy3uqDadKl6hA_2AiMXjqifbkMilwaLSAdpERL7b9NlcFeLE9ketVkXta3MIfqEcYK3LeK6qEYu7Ti6RAb-TL5YbJQd8bVLhuN_5D9WVgAsvKnEUSwZS4ZF2A1lHbUNf1L8pgQrfvq6U3XdZEWdUmdKOiTFaMwj99BjqBFEFkTpMel2BNM06rmUObl2rRwWUJcxrpIytD-VOcNcOUDB_SpadwJQtQJ43BumfeOD3a_v4l2IcyXXCrR2ocWplVTYMZK3N--u2sd7sFr_Qo8yJx_yjDZ2DYm-34WEkFv8QDGCDrmO-RNKR_n9kL29_ab7-_qcXEnDVtv6tw7SSO9676-xzfV_FgvBIill_NOSJky4wBc7HXFjfBOP7-bpoI5vpLz3nhjcX_sZot0u729jxc6FG_FhYJN7Cgf-QHwM4h1HyHWFzWNA0gnAIrl0UK-g0cZ42-jDbreTo50STc4-qI8N9_8yb_X-KyZYtPMPX2jw6rdh7qJE6B2kRNs0LX6Gus4BBCLciHbWnUH5efanQhLH75PpKam5BG_CSNlgfO9LPUM3lf5TGQfP4Lqm9YxzMsG7rFiTc6q3N2jtWam_cnis_v5PJo2eAUpWXNq6CVDcyPxLt9buGwWCkgn3vscDuvcPcW3ycE9xyWmyQYRQliXz5JuTWSTmFi9wi8TtkVH12en7jnLu2eBmK9JkbpSF08T6ZKxeyZP5imLDe5pAxrC1uyyf0v2L-mmChWV920jkVPwlZb3i-z9P-jBt)

Explication :

Ce diagramme de séquence illustre le processus d'exécution d'une sauvegarde dans l'application EasySave. Il met en évidence les interactions entre l’utilisateur et les différentes classes impliquées, en détaillant chaque étape du traitement.

L’utilisateur sélectionne l’option "Exécuter une sauvegarde" dans l’interface utilisateur (IHM). Cette action est envoyée à MenuController, qui gère les interactions principales du menu.

MenuController demande à BackupController d'afficher la liste des sauvegardes existantes. BackupController récupère les sauvegardes et les envoie au MenuView, qui les affiche à l’utilisateur. L’utilisateur est invité à sélectionner une ou plusieurs sauvegardes en entrant un numéro ou une plage de numéros (1-3 pour exécuter les sauvegardes 1 à 3).

Une fois l’index de la sauvegarde saisi, il est envoyé à BackupController, qui :

* Vérifie que la sauvegarde sélectionnée existe bien.
* Démarre le chronomètre pour mesurer le temps d’exécution.
* Contacte BackupStateJournal pour calculer l’état initial de la sauvegarde (nombre total de fichiers, taille totale, progression initiale).
* Demande à BackupStrategyFactory la stratégie de sauvegarde adaptée (Complète ou Différentielle) :

Cas 1 : Sauvegarde Complète

* Tous les fichiers sont copiés.
* BaseBackupStrategy exécute la sauvegarde en copiant tous les fichiers du répertoire source vers la destination.
* Une confirmation est renvoyée à BackupController une fois la sauvegarde terminée.

Cas 2 : Sauvegarde Différentielle

* Seuls les fichiers modifiés depuis la dernière sauvegarde sont copiés.
* BackupController récupère la date de la dernière sauvegarde à partir des logs (Logger).
* DifferentialBackupStrategy compare les fichiers pour identifier ceux à copier.
* BaseBackupStrategy exécute la sauvegarde en copiant uniquement les fichiers mis à jour.

Cas particulier : Si le répertoire cible n’existe pas

* Si aucune sauvegarde antérieure n’a été trouvée, une sauvegarde complète est automatiquement déclenchée.

À chaque fichier copié :

* BackupStateJournal met à jour l'état de la sauvegarde.
* Les observateurs (Observer Pattern) sont informés de la progression.
* ConsoleView affiche l’avancement en temps réel.
* Une pause (Thread.Sleep) est ajoutée pour que la progression soit visible.

Une fois la sauvegarde terminée :

* BackupController arrête le chronomètre et enregistre le temps total d’exécution.
* BackupStateJournal met à jour l’état de la sauvegarde en COMPLETED.
* LogController enregistre les informations de la sauvegarde dans un fichier journalier JSON via la DLL EasySave.Logger.

Les informations stockées incluent :

* Le nom de la sauvegarde
* La source et la destination des fichiers
* La taille totale des fichiers transférés
* Le temps de transfert

BackupController informe l’utilisateur que la sauvegarde s'est terminée avec succès. L'utilisateur est redirigé vers le menu principal.

1. Diagramme de séquence pour la consultation de l’historique :

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

Le contenu généré par l’IA peut être incorrect.

Lien vers le diagramme de séquence pour la consultation de l’historique :

[Online FlowChart & Diagrams Editor - Mermaid Live Editor](https://mermaid.live/edit#pako:eNp9VNtu2zAM_RVCL00AZx_ghwJFu3Ubkm5Y1z4MAQrGYmytsuTpErQo-kH9jv7YKNu5OUb9YpsizyEPKb6IwkoSufD0L5Ip6Eph6bBeGuAHi2Ad3HlygB7ugtLKY6DouuMGXVCFatAE-PZ1kXz4dXq2IBMvrQnOat1BHVtOI661XaG-R6dwpcmnkIHpNGZuy7JD778mV_P5tPNrS5hN2yxzuKys8srBUvx2aLzmkoCz8VEHdkt82pYe_troDGpFzi_FThC1Se4JcGDalZ5ImOu4xhwWaLCkC3Znqsn0A19OsU-tphActRzWQNHmTSfJjKk5QGSOgYB5kumL0vQTQ7VNZwc5qvawK7PZ-Xkvdg7XFHrAG6xpMoUJNg0xOUgCjbC2pqtCUhMVS8wHWve8kj5m7jvKVWzpPj8pH1hGuH9_c2rNTQJKpjTDXftkbDvYF6bbAWGr8o01ag99CJ_qGbatjeK_uHl_Gwtphyo_mqVDelhh8RibB7Y9PPMzWyxmUn76fvvjBibG1hlQwJCBZ9-CMlbCB2UwKZWBWTkINqBOGq5VUXGZnXX_Dx7jhkp08v0tg4CK086gsg6V449AdcPJnNETFTGhTrfjQ9rTaD1DBS5iEU0qaivDPmo4ZWNq7MP32gPGJI48q6La5WPk-OimrvSovygkTTFCzU7QOGXS7ddbjIM5GrsUB8e723pg6y61yERNrkYleS2-JKelCBXVfO9SChLdY6J7ZT-Mwd4-m0LkwUXKRGwk4_QrVORrZIkzQVLxGl10e7Zdt5lgIctq58H764-124jX_0gM5uo)

Explication :

L’utilisateur sélectionne "Consulter les logs journaliers" depuis l’interface utilisateur (IHM). L’IHM transmet cette action à MenuController, qui gère l’affichage du menu et l’exécution de l’action demandée.

MenuController envoie une requête à GlobalVariables pour récupérer le chemin du fichier log. GlobalVariables appelle la fonction GetLogFileName() de la DLL EasySave.Logger pour déterminer le fichier de log du jour.

Logger exécute la méthode Exists() pour vérifier si un fichier de log est disponible pour la date demandée. Deux cas de figure sont possibles :

*Cas 1 : Les logs sont disponibles :*

1. Logger confirme que les logs existent et envoie cette information à MenuController.
2. Logger récupère les détails du fichier de log, notamment :

* Nom du fichier (backup\_log\_yyyy-MM-dd.JSON)
* État de la sauvegarde (Active, Complétée, Échec…)
* Source et destination des fichiers
* Nombre total de fichiers traités
* Nombre de fichiers sauvegardés
* Taille totale des fichiers transférés
* Horaire de début et de fin de la sauvegarde
* Temps d’exécution

1. Ces informations sont envoyées à IHM pour être affichées à l’utilisateur.

*Cas 2 : Aucun log disponible :*

1. Logger informe MenuController qu’aucun log n’a été trouvé pour la date demandée.
2. MenuController affiche un message à l’utilisateur :

"Aucun log disponible aujourd’hui."

1. L’utilisateur est alors redirigé automatiquement vers le menu principal.

Après l’affichage des logs (ou en cas d’absence de logs), MenuController envoie une instruction à l’IHM pour retourner au menu principal. L’utilisateur peut alors choisir une autre action.

1. Diagramme de séquence pour quitter l’application :

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Lien vers le diagramme de séquence pour quitter l’application :

[Online FlowChart & Diagrams Editor - Mermaid Live Editor](https://mermaid.live/edit#pako:eNqNU0tOwzAQvYrlDVQKPUB2FRSBRCVQ6QZlM3KmrYUzTv2RqKpKXAc2HKI34SSM0zYKbRFdWLHjN-_NvPGspLIlylx6XEQkhTcaZg4qUdD3-5dIC1SwTkw8OgFeTII22kPA6DqYGlzQStdAQdzfjRIwff5AjJDitaXgrDFb1oM_p8MenW1SY_xuWyHvNKV7I0oUU3QVhuiww9DZNjVc9VJquRhvPgyqoC0Rw-VT1CGgK2QHn0pg-EFyOaOfHZBnqcBS0JAINbfaa_yfIBcjIJjhoInzl72u1yaIh4vY8VhZmmquqgPar4O8TggN31DFsFNiIZfMIRFcPI9vZ3Mubltjyyjq1vxjjn2Prjg62Z3MGtR1XDKhT09GDlmg8bm2fF5sbRcGuHm0-UyGciO5am8N9vv9X36i8eck3moPiKKBpj1M2mgcPxCkkk8ykxVfgi55GFYF8buRYY4VNzTVUIJ7LWRBa8ZBDHa8JCXzZGQmY11yr3aDI_MpcJaZxFLz3Iy209UMWSadjbN5i-BH_WLtPmL9A4vmX78)

Explication :

Ce diagramme de séquence illustre le processus de fermeture de l’application EasySave. Il montre comment l’utilisateur interagit avec l’interface pour quitter l’application et comment le programme gère cette demande.

L’utilisateur sélectionne l’option "Quitter" dans l’interface utilisateur (IHM). L’IHM transmet cette action au MenuController, qui est responsable de la gestion du menu et des actions utilisateur.

MenuController appelle la fonction ManageActions() pour traiter la demande de l’utilisateur. Une boîte de dialogue ou un message de confirmation est affiché pour demander à l’utilisateur s’il veut vraiment fermer le programme. Deux scénarios possibles sont alors traités via une condition alt :

Cas 1 : L’utilisateur confirme la fermeture

* ExecuteAction() retourne true, validant la demande de fermeture.
* MenuController envoie une requête au Programme Principal pour exécuter la fermeture.
* Un message s’affiche pour indiquer à l’utilisateur d’appuyer sur "Entrée" afin de fermer définitivement la console.

Cas 2 : L’utilisateur annule la fermeture

* Si l’utilisateur choisit d’annuler, MenuController annule la demande de fermeture.
* Un message est affiché pour informer l’utilisateur que la fermeture a été annulée.
* L’IHM reste active et retourne au menu principal.

Si l’utilisateur confirme, le programme se termine et ferme l’application. L’interaction utilisateur prend fin.

1. Documentation de la Dynamic Link Library :



1. Documentation Utilisateur :



1. Documentation Technique :



1. Release note V0.1 :



1. Conclusion :

Ce premier livrable marque une étape essentielle dans le développement d’EasySave, en posant les bases d’un logiciel de sauvegarde efficace et structuré.

Grâce à l'implémentation des principales fonctionnalités telles que la création de travaux de sauvegarde, l’exécution de sauvegardes (complète ou différentielle), la gestion des logs journaliers et le suivi en temps réel des sauvegardes, nous avons respecté les exigences du cahier des charges tout en garantissant un code propre, maintenable et évolutif.

L'intégration d’une Dynamic Link Library (DLL) dédiée à la gestion des logs renforce l’architecture modulaire du logiciel et assure une meilleure réutilisation du code pour les futures versions.

Enfin, les différents diagrammes UML réalisés permettent d’avoir une vision claire de l’architecture logicielle et des interactions entre les composants du système.

Perspectives pour la suite :  
Si cette version satisfait les attentes, nous pourrons envisager l’évolution vers EasySave 2.0, avec notamment une interface graphique (WPF) pour améliorer l'expérience utilisateur et faciliter l'utilisation du logiciel.

En conclusion, EasySave 1.0 est une première version fonctionnelle et robuste, qui pose les fondations d’un logiciel de sauvegarde fiable et évolutif.