26/02/2025

PROJET GENIE LOGICIEL – Livrable 3

Diagrammes



GROUPE 4

HOFSTETTER Léo – VASSILENKO mark – SANGLIER arthur – DEPPNER Aurélien

Table des matières

[Introduction 2](#_Toc190851920)

[Diagramme de cas d'utilisation 2](#_Toc190851921)

[Diagramme d'activité 5](#_Toc190851922)

[Diagramme de classes 7](#_Toc190851923)

[Diagramme de séquence 11](#_Toc190851924)

[Conclusion 16](#_Toc190851925)

# Introduction

Avant de développer l'application, nous avons réalisé plusieurs diagrammes pour avoir une vue d'ensemble de ce que doit faire l'application.

Ces diagrammes permettent de mieux comprendre le système et facilitent la communication avec les parties prenantes du projet. Ils sont également essentiels pour assurer une documentation technique structurée et garantir la maintenance et l’évolution du projet.

# Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation représente l’ensemble des interactions possibles entre les utilisateurs et l’application. Il permet de visualiser les différentes fonctionnalités offertes par le système et d’identifier les rôles des différents acteurs.

Voici les fonctionnalités de notre application :

**Page principale :**

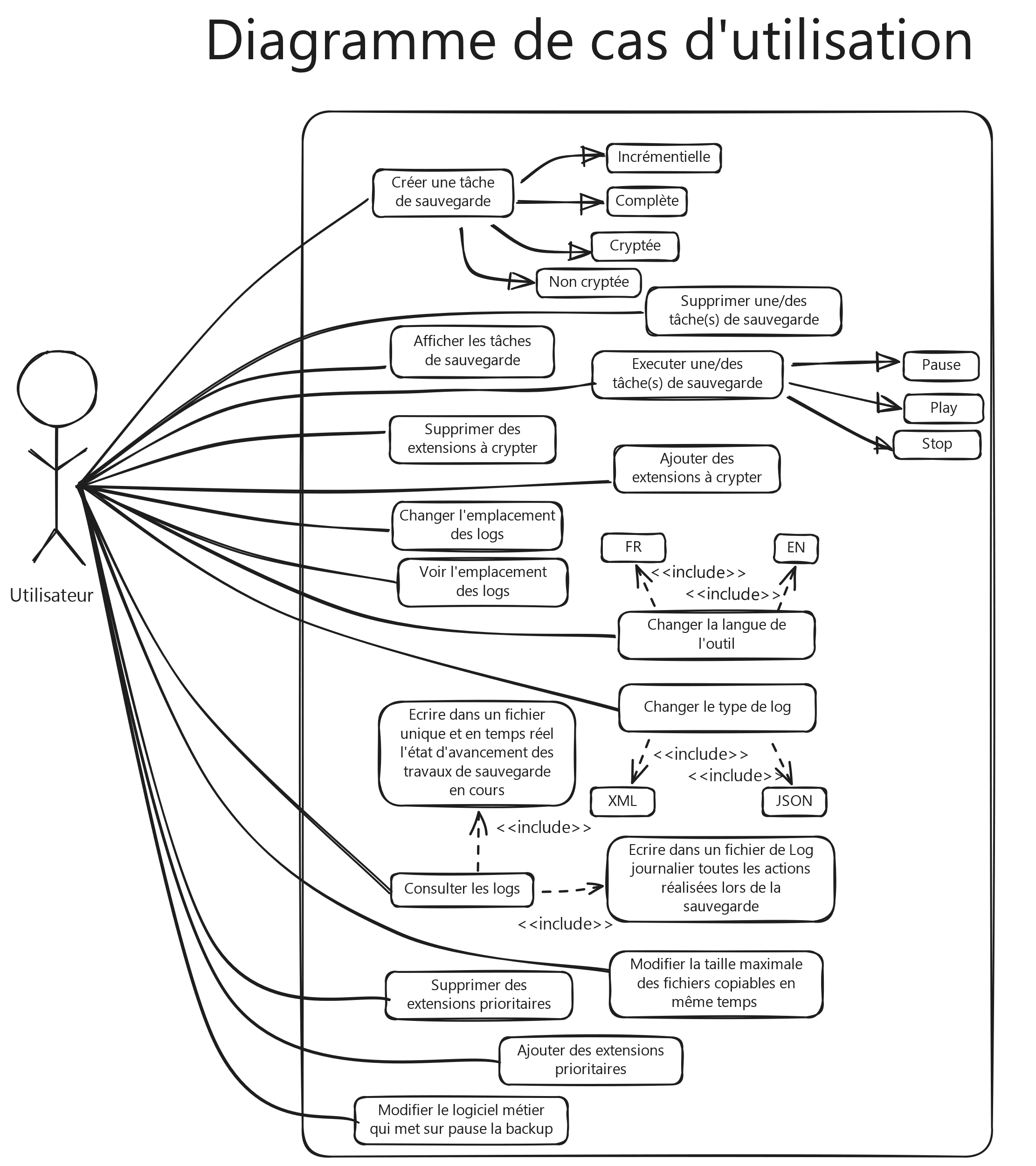
* Créer une sauvegarde : L'utilisateur définit un nom, un répertoire source et cible, un type de sauvegarde (complète ou différentielle), il peut aussi spécifier si la sauvegarde doit être cryptée. Si l’option cryptée est cochée, alors tous les fichiers ayant l’extension spécifié dans la partie paramètres seront cryptés. Si l’option différentielle est cochée, alors seulement les fichiers modifiés au cours de 24 dernières heures seront copiés.
* Afficher les sauvegardes : L’utilisateur peut afficher toutes les sauvegardes qui ont été créés. Il peut par ailleurs rafraichir la page en appuyant sur le bouton correspondant en bas à droite.
* Exécuter une sauvegarde : L'utilisateur sélectionne une sauvegarde existante et la lance. Il peut également décider d’exécuter plusieurs tâches de sauvegardes en maintenant Ctrl ou Maj.
* Mettre sur pause une sauvegarde en cours : Fini de copier le fichier en cours de copie, puis met immédiatement la ou les backup(s) en pauses.
* Mettre sur play une sauvegarde en pause : Si une ou plusieurs sauvegardes sont en pause, alors continue l’exécution des backup là où elles se sont arrêtées.
* Stopper une backup en cours d’exécution : Stop complètement et immédiatement une ou plusieurs backups qui sont en cours d’exécution. Si un fichier été en train d’être copié, celui-ci est supprimé du répertoire de destination.
* Supprimer une sauvegarde : L’utilisateur peut retirer une tâche enregistrée. Il peut également décider d’exécuter plusieurs tâches de sauvegardes en maintenant Ctrl ou Maj.

**Page paramètres :**

* Changer la langue : L'utilisateur peut modifier la langue de l'interface entre le français et l’anglais.
* Gérer les logs : L'application enregistre des informations sur les tâches exécutées dans des fichiers logs en temps réel et dans des fichiers logs journalier. Ces fichiers peuvent être consultés par l’utilisateur en appuyant sur le bouton correspondant en bas à droite.
* Modifier les types de logs : Choisir entre XML et JSON
* Modifier le répertoire de logs : Choisir le dossier dans lequel seront écrit les logs.
* Modifier le logiciel métier qui mettra sur pause l’exécution d’une sauvegarde : Permet à l’utilisateur de modifier le logiciel métier qui mettra sur pause l’exécution d’une sauvegarde.
* Ajouter et supprimer les extensions à crypter : Permet à l’utilisateur d’ajouter et de supprimer dans une liste, les extensions de fichier qui seront cryptées pour les sauvegardes cryptées.
* Ajouter et supprimer les extensions prioritaires : Permet à l’utilisateur d’ajouter et de supprimer dans une liste, les extensions de fichier qui seront copiées en priorité durant n’importe quelle sauvegarde.
* Modifier la taille maximale des fichiers à mettre en attente : Permet à l’utilisateur de modifier la taille en Mo avec un minimum de 10Mo, la taille à partir de laquelle les fichiers ne seront pas copiés en même temps, mais copiés à la fin pour laisser les plus petits fichiers être traités en priorité.

Ce diagramme est particulièrement utile pour la phase de spécification fonctionnelle et pour la présentation aux clients.

Voici notre diagramme de cas d’utilisation au format SVG :

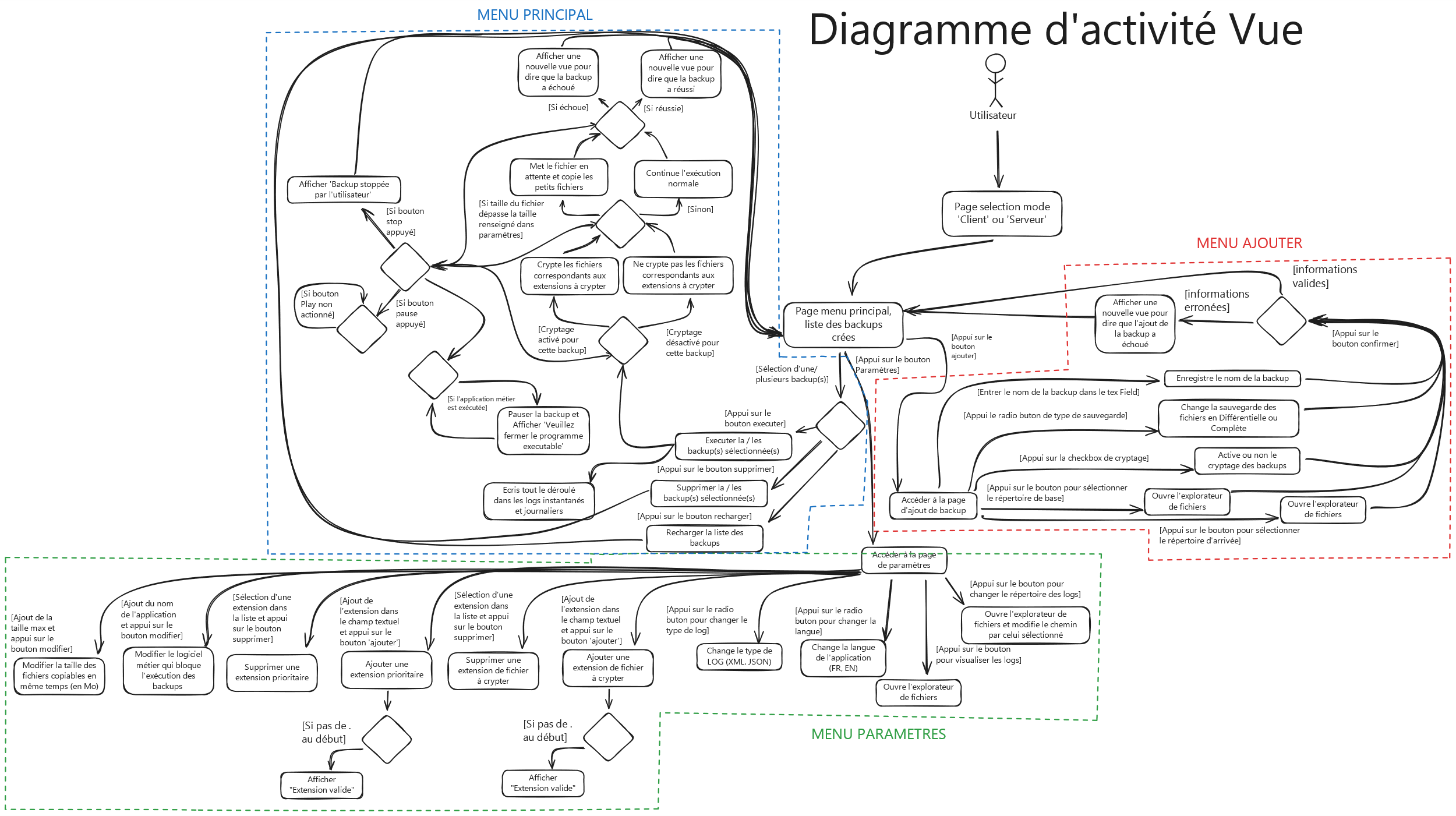


# Diagramme d'activité

Le diagramme d'activité permet de représenter le déclenchement des évènements en fonction de l'état du système. Il sert également à modéliser des processus pouvant s'exécuter en parallèle.

Dans notre projet, il montre le déroulement des différentes actions, comme la validation des paramètres avant la création d'une sauvegarde ou la mise à jour de l'état du système pendant l'exécution d'une sauvegarde.

Voici notre diagramme d’activité au format SVG :



# Diagramme de classes

Le diagramme de classes représente une architecture bien structurée, avec plusieurs couches assurant une séparation des responsabilités claire. Cette organisation facilite la maintenance, l’extension et l’évolution du projet.

**1. Couches de l'Architecture**

L’application est structurée en plusieurs couches bien distinctes :

1. Modèles (Models)

Les classes modèles représentent les données essentielles utilisées dans l’application.

* BackupModel : Représente une tâche de sauvegarde avec ses informations principales (nom, chemin source/destination, type de sauvegarde).
* BackupState : Stocke l’état d’une sauvegarde (en cours, terminée, erreur, etc.).
* FileTransferInfo : Contient des détails sur les fichiers transférés (chemin, taille, durée de transfert).

1. Contrôleurs (Controllers)

Les contrôleurs gèrent la logique métier et orchestrent les interactions entre les modèles et les services.

* BackupController : Responsable de la gestion des sauvegardes (ajout, exécution, suppression).
* LogController : Gère les logs et informe les observateurs en temps réel (Pattern Observer).
* LangController : Gère le changement de langue de l’application.

1. Services

Les services contiennent la logique principale de l’application et offrent des fonctionnalités avancées.

* BackupService (Implémente IBackupService) :
  + Gère l’exécution et le chargement des sauvegardes.
  + Responsable de la création et suppression des tâches.
* ConfigurationService :
  + Gère la configuration de l’application (fichier JSON).
  + Permet de charger et récupérer les paramètres.

1. Gestion des Fichiers

Les classes responsables du traitement des fichiers et de la gestion des transferts.

* FileController :
  + Assure la copie des fichiers (CopyDirectory, CopyModifiedFiles).
* File :
  + Offre des méthodes utilitaires pour copier et modifier des fichiers.

1. Gestion des Logs

L’application met en place une gestion avancée des logs.

* LogController :
  + Centralise la gestion des événements et erreurs.
* LogFileListener (Pattern Observer) :
  + Écoute les événements de logs et les écrit dans un fichier.
* RealTimeLogListener :
  + Stocke les logs en temps réel et permet leur consultation immédiate.

**2. Design Patterns Utilisés**

L’application repose sur plusieurs design patterns qui permettent d’avoir une architecture modulaire et évolutive.

1. Singleton

Pattern utilisé dans : ConfigurationService, LogController, BackupController, LangController.

* Objectif : Garantir qu’une seule instance de ConfigurationService existe dans toute l’application.
* Pourquoi ?
  + Évite la duplication des paramètres de configuration.
  + Assure une lecture cohérente des fichiers de configuration JSON.

1. Observer

Pattern utilisé dans : LogFileListener, RealTimeLogListener

* Objectif : Permet de notifier plusieurs observateurs lorsqu’un événement se produit (gestion des logs).
* Pourquoi ?
  + Facilite l’ajout de nouveaux observateurs sans modifier la logique principale.
  + Permet d’avoir des logs en temps réel et des fichiers journaux séparés.

1. Command

Pattern utilisé dans ICommand, CommandController et tous les fichiers comme ExecuteCommand …

* Objectif : Simplifier l’évolutivité des actions des boutons en séparant la logique.
* Pourquoi ?
  + Pour séparer la logique de la vue.

Exemple de fonctionnement :  
1️ LogController enregistre un événement de log.  
2️ Il notifie tous les observateurs (LogFileListener, RealTimeLogListener).  
3️ Chaque observateur traite l’information (ex : écrit dans un fichier log).

1. Facade

Pattern utilisé dans : ClientCommunicationStrategy, ServerCommunicationStrategy et communicationFacadeController

* Objectif : Fournir une interface simplifiée et unifiée qui masque la complexité d’un sous-système.
* Pourquoi ?
  + Le code devient plus lisible et organisé, car la logique est encapsulée dans la façade plutôt que dispersée dans plusieurs classes.
  + Toutes les interactions vers le sous-système passent par un point central, ce qui permet de mieux contrôler et modifier les comportements si nécessaire.

Voici notre diagramme de classes :

[Lien vers le diagramme :](https://www.mermaidchart.com/raw/e393131e-9715-4f6b-85bb-a8af3d847df3?theme=light&version=v0.1&format=svg)

# Diagramme de séquence

Le diagramme de séquence illustre les interactions entre les composants du système en suivant un ordre chronologique. Il permet de visualiser le flux des appels de méthodes et la communication entre les objets.

Voici les différents scénarios :

1️) Lister les sauvegardes

2️) Créer une sauvegarde

3️) Exécuter une sauvegarde

4️) Supprimer une sauvegarde

5) Mettre sur pause une sauvegarde

6) Mettre sur play une sauvegarde

7) Arrêter une sauvegarde

8) Changer la langue de l'application

9) Changer le format des logs

10) Changer le répertoire des logs.

11) Modifier le logiciel métier qui met sur pause l’exécution d’une Backup

12) Modifier la taille max d’une copie simultanée de 2 fichiers

**1️ Lister les sauvegardes**

Objectif : Afficher toutes les sauvegardes existantes.

Déroulement des actions :

1. L'utilisateur demande à voir la liste des sauvegardes.
2. JobController interroge BackupService pour récupérer toutes les sauvegardes.
3. BackupService charge les sauvegardes à partir du fichier de configuration (ConfigurationService).
4. Les informations sont envoyées à JobController.
5. L’interface utilisateur affiche la liste des sauvegardes disponibles.

**2️ Création d’une sauvegarde**

Objectif : Ajouter une nouvelle sauvegarde.

Déroulement des actions :

1. L’utilisateur entre les paramètres de la sauvegarde et valide.
2. JobController reçoit la requête et vérifie les informations.
3. BackupTask est créé avec :
   * Nom
   * Source
   * Destination
   * Type de sauvegarde (Complète ou Différentielle)
4. BackupSnapshot enregistre un instantané avant l’exécution.
5. SnapshotController sauvegarde l’instantané pour une future restauration.
6. LogController génère une entrée de log indiquant la création de la sauvegarde.
7. L’état du système est mis à jour.
8. Un message de confirmation est affiché à l’utilisateur.

**3️ Exécution d’une sauvegarde**

Objectif : Démarrer une sauvegarde existante.

Déroulement des actions :

1. L’utilisateur sélectionne une tâche existante et lance l’exécution.
2. JobController récupère les informations de la tâche et sélectionne la stratégie de sauvegarde (CompleteType ou DifferentialType).
3. FileController liste les fichiers concernés.
4. Les fichiers sont envoyés à BackupTask pour être copiés vers la destination.
5. LogController enregistre chaque étape
6. StateManager met à jour l’état en temps réel :
   * "En cours" → "Terminé"
7. Une notification est envoyée à l’utilisateur.

**4️ Suppression d’une sauvegarde**

Objectif : Supprimer une sauvegarde existante.

Déroulement des actions :

1. L’utilisateur demande la suppression d’une sauvegarde.
2. JobController vérifie que la sauvegarde existe.
3. BackupSnapshot crée un instantané de l’état avant suppression.
4. JobController demande à FileController de supprimer les fichiers concernés.
5. StateManager met à jour l’état et supprime la référence de la sauvegarde.
6. LogController ajoute une entrée de log pour indiquer la suppression.
7. Un message de confirmation est affiché à l’utilisateur.

**5 Mettre sur pause une sauvegarde**

Objectif : Mettre sur pause une ou plusieurs sauvegardes en cours d’exécution.

Déroulement des actions :

1. L’utilisateur demande la pause.
2. La méthode btnPauseBackup\_Click est exécutée
3. Elle renvoie true à la méthode TogglePauseExecution(bool pause)
4. La méthode rend invisible le bouton Pause et rend visible le bouton Play
5. Elle met sur pause chacune des backup.

**6 Mettre sur play une sauvegarde**

Objectif : Mettre sur play une ou plusieurs sauvegardes qui sont actuellement sur pause.

Déroulement des actions :

1. L’utilisateur demande le play .
2. La méthode btnPlayBackup\_Click est exécutée
3. Elle renvoie false à la méthode TogglePauseExecution(bool pause)
4. La méthode rend visible le bouton Pause et rend invisible le bouton Play
5. Elle met sur play chacune des backup.

**7 Stopper une sauvegarde**

Objectif : Stopper toutes les sauvegardes en cours d’exécution.

Déroulement des actions :

1. L’utilisateur demande le stop .
2. La méthode btnStopBackup\_Click est exécutée
3. Elle déclenche le `\_cancellationTokenSource`
4. Elle affiche un popup pour dire que les backups sont stopées.
5. Celui-ci est envoyé en paramètre à la fonction ExecuteBackup de la classe BackupController
6. Une condition stop l’écriture des Logs et écrit `Canceled` dans ceux-ci.
7. Le `\_ cancellationTokenSource`est envoyé en paramètre à la fonction Exécute de la classe BackupController.
8. Cette fonction envoie en paramètre le Token à la méthode CopyFiles de la classe fileController.
9. La méthode CopyFiles regarde en permanence si le Token a été déclenché, si tel est le cas alors il arrête la boucle de copie des fichiers et passe en paramètre ce token à la méthode CopyFile de la méthode FileModel
10. Cette méthode déclenche une condition et supprime le fichier du répertorie de destination si celui-ci n’a pas été copié entièrement.

**8 Changer la langue de l’application**

Objectif : Modifier la langue de l’interface.

Déroulement des actions :

1. L’utilisateur sélectionne une nouvelle langue dans les paramètres.
2. LangController reçoit la requête et met à jour la configuration.
3. ConfigurationService enregistre la nouvelle langue dans le fichier JSON de configuration.
4. Tous les éléments de l’interface sont mis à jour pour refléter la nouvelle langue.
5. Un message de confirmation est affiché.

**9 Changer le format des logs**

Objectif : Modifier la façon dont les logs sont enregistrés (ex : texte, JSON, XML).

Déroulement des actions :

1. L’utilisateur choisit un nouveau format de logs dans les paramètres.
2. LogController met à jour la configuration du format.
3. ConfigurationService enregistre le nouveau format dans le fichier JSON.
4. Tous les nouveaux logs seront désormais enregistrés dans le nouveau format choisi.

**10 Changer le repertoire des logs**

Objectif : Modifier le répertoire où les logs sont enregistrés.

Déroulement des actions :

1. L’utilisateur choisit un nouveau répertoire de destination dans les paramètres.
2. LogController met à jour le répertoire des logs.
3. La modification est enregistrée dans Settings.settings
4. Tous les nouveaux logs seront désormais enregistrés dans le nouveau répertoire choisi.

Nous avons réalisé le diagramme de séquence pour la fonctionnalité permettant d’ajouter une sauvegarde.

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

Le contenu généré par l’IA peut être incorrect.

Nous avons aussi réalisé un diagramme de séquence se concentrant uniquement sur l’exécution d’une save et l’enregistrement des logs associés.   
  
[Voici le lien :](https://mermaid.live/view#pako:eNrlV91u4kYUfpXRSKsFyaGQJaFBykrBKdWucBoVUqSKm4l9gGnGM-7MOAuJ8i695Tl4sZ6xDeHHpHR3e1VHisz483fOfOdnjp9pqCKgbWrgzxRkCNecTTSLR5LglTBtecgTJi25I8yQO8sFN8xCqvcRwdBBAsblkMtIfdlH-F2H8FUcd1nIIthHdHyH6LDwIU18Ja1WQkCJrU7wigtwB2If0suoemryFk-370BdLqA_NxZw3znm3Tt0U475JNXMciVJBIYINTGkohK3IAEJPcIeHYt4D7PlIkzdg2pOIJUFoh5Box_tA1w5sueffPzoQH2w6O1gnkBlRD_3f7kZ0Wop5JprCK3Sc8SFU4i5XCPz_3cID4ZoVvCQmFSTEf1pBugfFJKNaA4MhgXyt-VC8zEHIhgxy4VA_szT9xxFKRxlwpKrNEwlcYsbMLlc5IgtyqvxmKN3JABj2AQ6aobu3qhP-G4f3JsQrd0GYWCHwu-67croysxlWHndwH2xgWqJSV8vF0B8hnksRKb1QD2A7KtUh0DAEi655QxTGEii1USjax2md5i6_TZZS1z7GaxLD1MxGcuu2Q5GpXAtl7YiVewR6-x6JE0iLJbbwpRHwqlSBh3iT5tEDFV8ZC44_uuiuzp-YaGLBfVKXy1DBWs_Kv_K-LYDwfaDjD4oNDmsxOrq9k8KV3ocqwkTnbgc4KDNNjiCgybRXLHpX8GiIek4BPgqmQdgNQ93qIRSCblFIO6PYQ8j8XKBMLwr2cpKz3W6h7u5sv9SlvZSpmJVvDGTEebZPnJtwpUq1mkesTwq-GqlVqt5WIpFfm6k_57JlTg9vxyw3RTYI6A1U-Hmms3d3eWYYUFV_-FdF1MX0NpQY0FfCTGAma1gY3KLt8xOPWJAu3J5gsjRHiDcCOYhf1GVokQzz_ZBrgGQA2Kg_ngy6XnWdwc8BnJ5SU4a9frh_R0RhU_SgtZpgm2IfOF2Sv5Q94QlycGgHB2Ybw3Odw_QkUHaDpTV6YEUfyNWR2u_6kr_e7UPSYXuV_Kj7obFOGjkPddzs4PlMmtFXtYYXTvP7waaSTMGbV2NeDs181_L7NLl61WOMpYyoQv-b9Max4iSlrO7uLdwRCbjwczNtKSVv-nT1-Trd8nRN7Tarvwfdvr05oHtlw171qJ8GsiAmYfacAoSnSvB4RFOln9hr8XzemMCc7OZuBfrrnDkNLn6TogTAdk4SVRKVstdxjfP2Cy61KMx6JjxCD95nt2DEbU4QcOItvE2YvrBjcYviGOpVX0cPWnb6eHRfJoqPo9Wi_gN8btSmz9p-5nOaPus1aidNvDvrH7aajWaFx6d03a91jhvnp03642z00ajWW_WWy8efcoY6rWLi_MmLv_onrU-fGh5FCKOM2iQf6FlH2oe1SqdTGk7i8_L36737GI)

# Conclusion

Ces diagrammes forment une documentation essentielle pour le développement du projet. Ils permettent :

* Une meilleure compréhension des fonctionnalités du système.
* Une visualisation claire des interactions entre les composants.
* Une application des bonnes pratiques de conception avec des Design Patterns robustes.

Cette approche assure également une évolutivité et une maintenabilité optimales du projet.