|  |
| --- |
| Récupération des AutoEvaluations |

Table des matières

[1 Analyse préliminaire 3](#_Toc129962220)

[1.1 Introduction 3](#_Toc129962221)

[1.2 Objectifs 4](#_Toc129962222)

[1.3 Planification initiale 4](#_Toc129962223)

[2 Analyse / Conception 5](#_Toc129962224)

[2.1 Concept 5](#_Toc129962225)

[2.2 Gestion de projet 6](#_Toc129962226)

[2.2.1 Méthodologie de travail 6](#_Toc129962227)

[2.2.2 Outil de versioning 7](#_Toc129962228)

[2.3 Stratégie de test 8](#_Toc129962229)

[2.4 Risques techniques 8](#_Toc129962230)

[2.5 Planification 8](#_Toc129962231)

[2.6 Dossier de conception 9](#_Toc129962232)

[3 Réalisation 11](#_Toc129962233)

[3.1 Dossier de réalisation 11](#_Toc129962234)

[3.1.1 Environnement 11](#_Toc129962235)

[3.1.2 Version du projet 11](#_Toc129962236)

[3.1.3 Fichiers de configurations 12](#_Toc129962237)

[3.1.4 Fichiers de scripts 14](#_Toc129962238)

[3.1.5 Librairies externes 24](#_Toc129962239)

[3.2 Description des tests effectués 24](#_Toc129962240)

[3.3 Liste des documents fournis 25](#_Toc129962241)

[3.4 Glossaire 26](#_Toc129962242)

[4 Conclusion 26](#_Toc129962243)

[5 Annexes 28](#_Toc129962244)

[5.1 Résumé du rapport du TPI / version succincte de la documentation 28](#_Toc129962245)

[5.2 Sources – Bibliographie 28](#_Toc129962246)

[5.3 Journal de travail 31](#_Toc129962247)

[5.4 Planification détaillée 39](#_Toc129962248)

[5.5 Manuel de modifications 48](#_Toc129962249)

[5.5.1 Modification des scripts 48](#_Toc129962250)

[5.5.2 Compilation des scripts 49](#_Toc129962251)

[5.5.3 Modifications du modèle d’auto-évaluations 50](#_Toc129962252)

[5.6 Manuel d'Utilisation 55](#_Toc129962253)

[5.6.1 Téléchargement 55](#_Toc129962254)

[5.6.2 Configurations 56](#_Toc129962255)

[5.6.3 Création des auto-évaluations 57](#_Toc129962256)

[5.6.4 Rapatriement des auto-évaluations 58](#_Toc129962257)

[5.7 Archives du projet 61](#_Toc129962258)

# Analyse préliminaire

## Introduction

Actuellement les enseignants en informatique utilisent un système d’évaluation des projets d’élèves, qui a été réalisé avec Excel plus des macros. Ce système offre une fonctionnalité d’exportation des onglets des élèves dans des fichiers Excel sans macro, pour leur permettre de s’auto-évaluer.

Toutes ces manipulations sont effectuées à la main et prennent beaucoup de temps. Cela est donc propice aux erreurs notamment lors du rapatriement des évaluations qui est fait en copier-coller.

Le but de ce projet, est de proposer une solution d’automatisation la plus complète possible, afin de minimiser le travail requis ainsi que les erreurs.

La technologie qui a été choisie est PowerShell. Car étant un langage de script natif à Windows, les manipulations afin d’installer le projet sont minimes. De plus, le langage étant développé par Microsoft, il y’a de nombreux composants natifs permettant d’interagir avec les différents programmes / processus Windows. Notamment Microsoft Excel.

Le projet se déroule dans le même cadre que lors d’un TPI. L’environnement logiciel et matériel est le suivant : 1 PC standard de l’ETML, Visual Studio Code et un dépôt GitHub.

Les différents modules qui seront utiles au projet sont les suivants : 122-ScriptsMacros, 158-GererMigrationLogiciels, 226-Prog-ImplementerOO, 302-BureautiqueAvancee, 306-Projet-RealiserPetitProjet, 403-Prog-Structuree, 404-Prog-BaseeObjets

La méthodologie de projet qui sera utilisée, est la méthode des 6 pas. La taille du projet étant assez légère une méthode agile est trop volumineuse à mettre en place.

Le projet s’effectuant seul, la méthode des 6 pas parait une des plus simple à utiliser. De plus c’est une méthode que nous avons vu en module. Nous sommes donc à l’aise dans son utilisation.

## Objectifs

Les objectifs techniques du projet définis dans le cahier des charges, sont les suivants :

1. Les valeurs proposées par les élèves sont récupérées correctement

2. Les commentaires des élèves sont récupérés correctement

3. Toutes les valeurs des indicateurs sont récupérées correctement

4. Le dispositif est pertinent et simple à utiliser

5. Le dispositif est fiable. En cas d’incidents, les messages d’erreurs sont pertinents

6. Une modification de la structure de la grille est possible

7. Fonctionne sans autorisation spéciale, ni configuration exigeant l’intervention d’un spécialiste système.

## Planification initiale



Figure Planification initiale

La planification initiale du projet a été élaborée en suivant méthode des 6 pas. Ainsi, le projet a été découpé en semaines, chacune de ces dernières étant subdivisée en quatre jours de travail. Ce qui correspond aux jours de projet.

# Analyse / Conception

## Concept

Le concept de ce projet se résume en l’automatisation de la gestion des auto-évaluations. Ces auto-évaluations sont utilisées pour que les élèves se notes eux même selon ce qu’ils ont accompli durant les projets.

Voici à quoi va ressembler le projet.

* La première étape consiste au remplissage des données du projet par l’utilisateur. Par exemple, nom du projet, durée… Ainsi que la liste des élèves.
* Ensuite l’utilisateur lance le script puis choisi l’option de création des auto-évaluations. (1)
* Le script crée ensuite une auto-évaluation par élève.
* L’utilisateur doit ensuite envoyer les auto-évaluations aux élèves.
* Les élèves remplissent les auto-évaluations et l’envoie ensuite à l’enseignant.
* L’utilisateur relance le script puis choisis cette fois l’option de rappariement des auto-évaluations. (2)
* Le script rapparie ensuite toutes les auto-évaluations puis crée une synthèse.
* Pour finir, l’utilisateur corrige les auto-évaluations et les renvoie corrigées aux élèves.

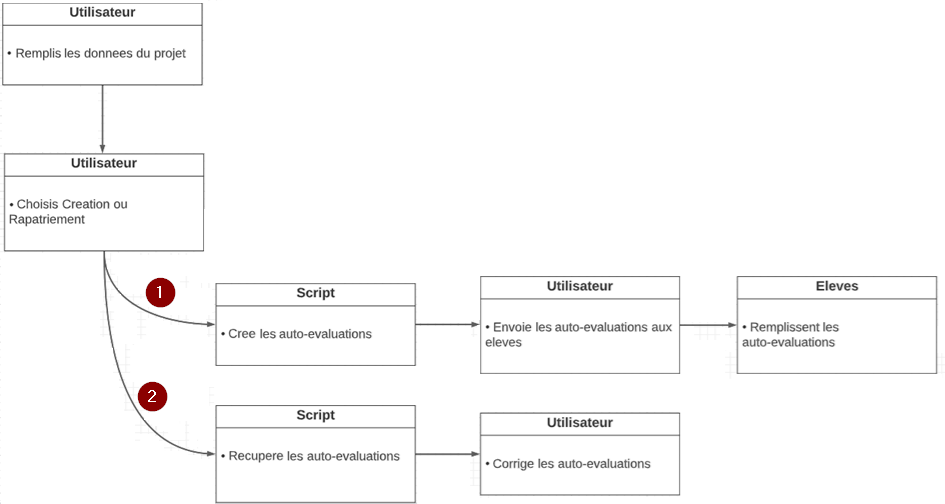


Figure Schéma de fonctionnement

## Gestion de projet

### Méthodologie de travail

La méthodologie que nous avons utilisé pour ce projet se nomme "La méthode des 6 pas/étapes".

Nous avons choisi cette méthode, car elle est relativement simple à mettre en place et en vue de l’envergure du projet et du nombre de personnes impliquées elle est efficace. De plus nous avons beaucoup utilisé cette méthode lors des différents projets tout au long de notre formation.

Voici en quoi consiste cette méthode.

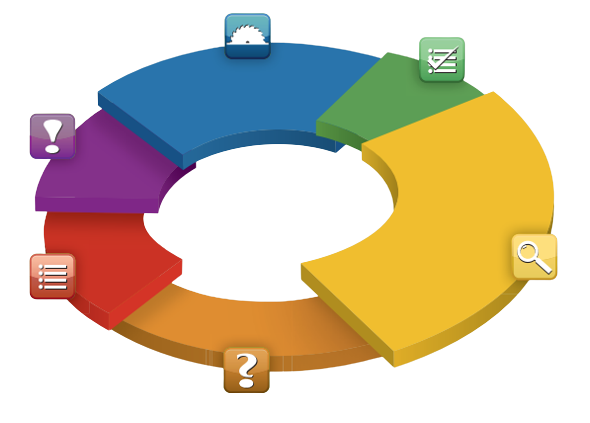


Figure Schéma méthode des 6 pas



#### 1. S'informer

La première étape consiste à comprendre et à enregistrer la tâche, à bien cerner le sujet et à pouvoir se faire une idée de l’objectif. Se procurer systématiquement des informations est, à ce stade, d’une

Importance capitale.

#### 2. Planifier

La planification est le moyen d’arriver au but. Pour cela, il faut l’imagination nécessaire pour mettre de l’ordre dans les opérations. Sur le plan du travail, il convient de choisir les outils et moyens auxiliaires, d’estimer les temps nécessaires et de fixer, pour les travaux en équipe, la répartition des tâches ainsi que les critères de contrôle et d’évaluation.

#### 3. Décider

Lorsque l’information et la planification ont été effectuées de manière optimale, il est plus difficile, en règle générale de prendre une décision. Lorsqu’on retarde des décisions, cela coûte beaucoup de temps et au pire des cas, il n’y a plus rien à décider, parce qu’on se trouve confronté à des contraintes matérielles. En équipe ou en petit groupe, il faudrait toujours, dans la mesure du possible prendre des décisions consensuelles (droit reposant sur un consensus qui a mis d’accord la grande majorité).

#### Une image contenant texte, outil Description générée automatiquement4. Réaliser

La réalisation ou exécution peut représenter la partie principale d’une tâche, comme c’est le cas pour les tâches de production, ou ne représenter qu’une petite partie de cette dernière lorsqu’il s’agit, par exemple, de l’organisation d’une manifestation. Le plan de travail établi au cours de l’étape de planification doit en principe être respecté et ne devrait pas être modifié à la légère.

#### 5. Contrôler

Chaque travail effectué doit être contrôlé avant d’être confié à des tiers. Contrôler signifie, par exemple, relire encore une fois, recalculer, comparer avec les prescriptions du cahier des charges, mesurer, estimer, etc. Un contrôle apporte de bons résultats lorsqu’il est effectué d’une autre manière ou par une personne différente de celles qui ont été employées pour faire le travail.

#### Une image contenant texte, clipart Description générée automatiquement6. Évaluer

Au moment de l’évaluation, nous revoyons éventuellement avec l’employeur la tâche exécutée. A cet effet, on passe encore une fois en revue le déroulement des opérations et on cherche à savoir ce qui a bien fonctionné et ce qui s’est moins bien passé et pourrait être amélioré la prochaine fois.

### Outil de versioning

L’outil de versioning que nous avons utilisé, est Git avec la plateforme GitHub.

Vous trouverez le répertoire avec le lien suivant : <https://github.com/SylvainPhilipona/Recuperation-des-AutoEvaluations>

Le répertoire à été organisé en deux branches. ‘’Main’’ et ‘’Dev’’. Toutes les fonctionnalités sont développées dans la branche ‘’Dev‘’, puis une fois fonctionnelles, la branche est ‘’merged’’ dans la branche ‘’Main‘’.

## Stratégie de test

|  |  |
| --- | --- |
| Test | Impact en cas d’échec |
| Le script démarre et affiche l’interface | Impossibilité d’utiliser le projet. |
| Les auto-évaluations sont créées avec les données spécifiées dans le fichier de configuration | Impossibilité d’utiliser le projet. |
| Les résultats des auto-évaluations sont correctement récupérés | Synthèse des notes impossible |
| Les commentaires des élèves sont correctement récupérés | Correction par l’enseignant biaisée |
| Les auto-évaluations sont correctement exportées en PDF | Envoie des notes corrigées doit être fait à la main |
| Aucun droit ni configuration nécessitant un spécialiste n’est nécessaire | Utilisation limitée selon les personnes |
| Fonctionnement sur Windows 11 | Le projet va très vite être obsolète |

## Risques techniques

Les risques techniques de ce projet, sont surtout liés à la compatibilité de la machine utilisé par l’utilisateur. Le projet ne devant pas nécessiter l’intervention d’un technicien pour installer le projet.

La solution est d’automatiser l’installation des modules lors de la première exécution du programme. De cette manière l’utilisateur n’a rien à faire et le programme installe tout ce dont il a besoin.

## Planification

Pour ce projet, la méthodologie de projet qui sera utilisée est la méthode des 6 pas.

Ce projet se déroule en 9 semaines. Du Lundi 23 Janvier 2023 au Vendredi 24 Mars 2023. Cela correspond à 202.5 heures.

Les semaines durent 22.5 heures sont découpées de la manière suivante.

Lundi -> 6h, Mardi -> 0h, Mercredi -> 6h45m, Jeudi -> 3h, Vendredi -> 6h45m

Les semaines ont ensuite été découpées selon les différentes étapes de la méthode des 6 pas. Tout d'abord, la collecte d'informations nécessaires pour le projet dans l'étape "Informer". Ensuite, l'étape "Planifier" qui implique l'organisation de toutes les autres étapes du projet. Puis, l'étape "Décider" pour choisir les outils et les moyens à utiliser pour le projet. Ensuite, vient l'étape "Réaliser" pour la mise en œuvre du projet. L'étape suivante, "Contrôler" ou "Tester", permet de vérifier que les fonctionnalités du projet sont opérationnelles. Enfin, l'étape "Evaluer" sert à vérifier que toutes les attentes du projet ont été correctement satisfaites. La méthode des six étapes est facile à appliquer et peut être utilisée aussi bien pour des projets professionnels que personnels.

La planification détaillée se trouve au point **5.4**

## Dossier de conception

Pour réaliser ce projet, nous avons utilisé un ordinateur standard de l’ETML sous le système d’exploitation Windows 10 Éducation. De plus nous avons utilisé le logiciel Visual Studio Code avec l’extension PowerShell pour le développement.

Pour l’utilisation du projet, il n’y aura pas besoin d’outils logiciels autre que ceux installés par défaut sur les PC. (Office365, explorateur de fichier, etc.)

La structure de fichiers se compose de la manière suivante :

* Un dossier ‘’Scripts‘’ contenant tous les scripts nécessaires ainsi qu’un dossier ‘’01-config‘’ contenant les fichiers de configuration et de modèle.
* Un fichier ‘’Compile-Project.ps1‘’ permettant de compiler le projet.

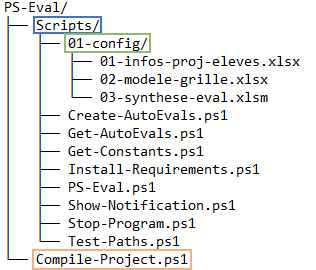


Figure Arborescence du projet

Voici la structure de code du projet :

* **PS-eval.ps1 :** affiche une interface graphique, puis permet à l’utilisateur de lancer la création et le rapatriement des auto-évaluations. Une notification est affichée pour avertir que les scripts peuvent prendre un certain temps à s’exécuter.

Pour la création, le script appelé est ‘’Create-AutoEvals.ps1 ‘’

Pour le rapatriement, le script appelé est ‘’Get-AutoEvals.ps1‘’

Pour l’affichage de la notification, le script appelé est ‘’Show-Notification.ps1‘’

* **Show-Notification.ps1 :** affiche une notification Windows pour l’utilisateur avec un message défini en argument.
* **Create-AutoEvals.ps1 :** crée les auto-évaluations des élèves selon les données spécifiées dans le fichier ‘’01-infos-proj-eleves.xlsx‘’ ainsi que selon le modèle ‘’02-modele-grille.xlsx‘’.

Pour installer les modules nécessaires, le script appelé est ‘’Install-Requirements.ps1‘’

Pour vérifier que les fichiers nécessaires existent bien, le script appelé est ‘’Test-Paths.ps1‘’

Pour obtenir les constantes, le script appelé est ‘’Get-Constants.ps1‘’

Pour arrêter le programme en cas d’erreur, le script appelé est ‘’ Stop-Program.ps1‘’

* **Get-AutoEvals.ps1 :** rapatrie les auto-évaluations qui ont été remplies par les élèves. Puis ce script crée une synthèse des évaluations dans un nouveau fichier.

Pour installer les modules nécessaires, le script appelé est ‘’Install-Requirements.ps1‘’

Pour obtenir les constantes, le script appelé est ‘’Get-Constants.ps1‘’

Pour arrêter le programme en cas d’erreur, le script appelé est ‘’ Stop-Program.ps1‘’

* **Test-Paths.ps1 :** Teste si plusieurs chemins de fichiers / dossiers donnés en arguments existent. Retourne une liste avec tous les fichiers / dossiers non existants.
* **Install-Requirements.ps1 :** Installe automatiquement les modules nécessaires au bon fonctionnement des scripts
* **Stop-Program.ps1 :** Arrête l’exécution du script qui l’appelle et retourne une erreur spécifiée en argument
* **Get-Constants.ps1 :** Contient toutes les constantes, et les retourne en tant qu’objet.
* **Compile-Project.ps1 :** Récupère le contenu de tous les scripts mentionnés ci-dessus, et les compile en un seul. Cela permet d’utiliser le projet avec plus de facilitée.

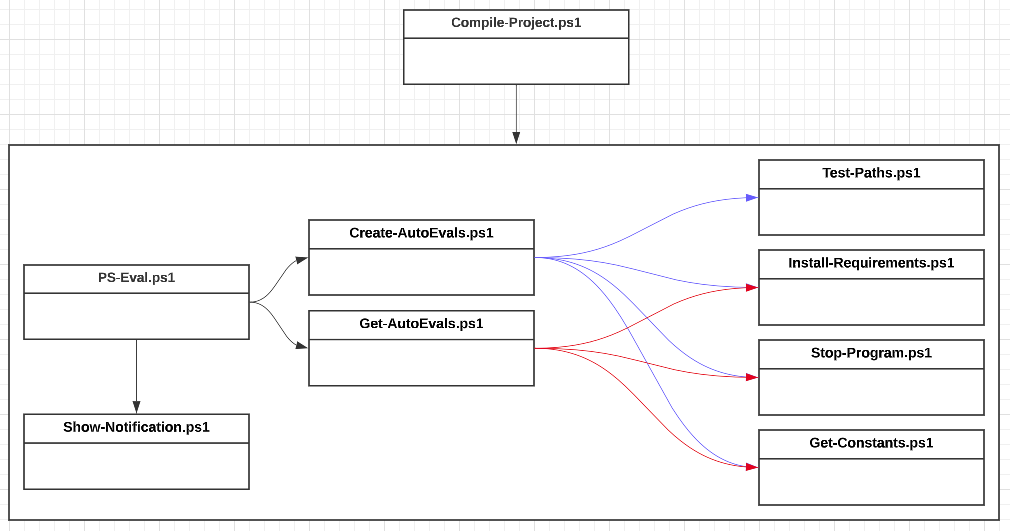


Figure Schéma de fonctionnement du projet

# Réalisation

## Dossier de réalisation

### Environnement

Pour ce projet, nous avions à disposition un ordinateur standard de l’ETML. Les spécifications sont les suivantes :

* Windows 10 Education 21H2 64 Bits
* Intel i7-11700 2.50Ghz
* RAM 32 Gb 3200 MHz

L’environnement logiciel est le suivant :

* Visual Studio Code, Version 1.76.1
* GitHub Desktop, Version 3.2.0
* PowerShell, Version 5.1.19041.2364

### Version du projet

Ce projet est en version 1.0. Car nous sommes partis de 0 pour le réaliser.

### Fichiers de configurations

Voici les différents fichiers de configuration que nous allons aborder :

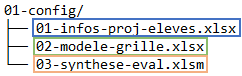


Figure Fichiers de configuration

**01-infos-proj-eleves.xlsx :**

Ce fichier contient la liste des élèves ainsi que les différentes informations du projet.

Les champs doivent être remplis par l’enseignant avant de lancer création des auto-évaluations.

Ces valeurs seront ensuite utilisées pour la génération des auto-évaluations.

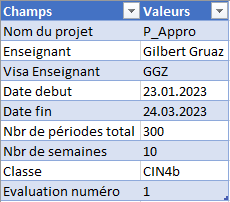
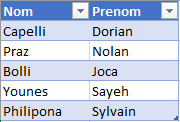


Figure Configurations des auto-évaluations

Figure Liste des élèves

**02-modele-grille.xlsx :**

Ce fichier est le modèle d’auto-évaluations qui est actuellement utilisé par les enseignants. Les données de titre (en rose) sont les champs qui seront remplacés par les valeurs définies dans le fichier ‘’01-infos-proj-eleves.xlsx‘’

La feuille Excel est verrouillée contre les modifications, sauf les cellules encadrées en rouge

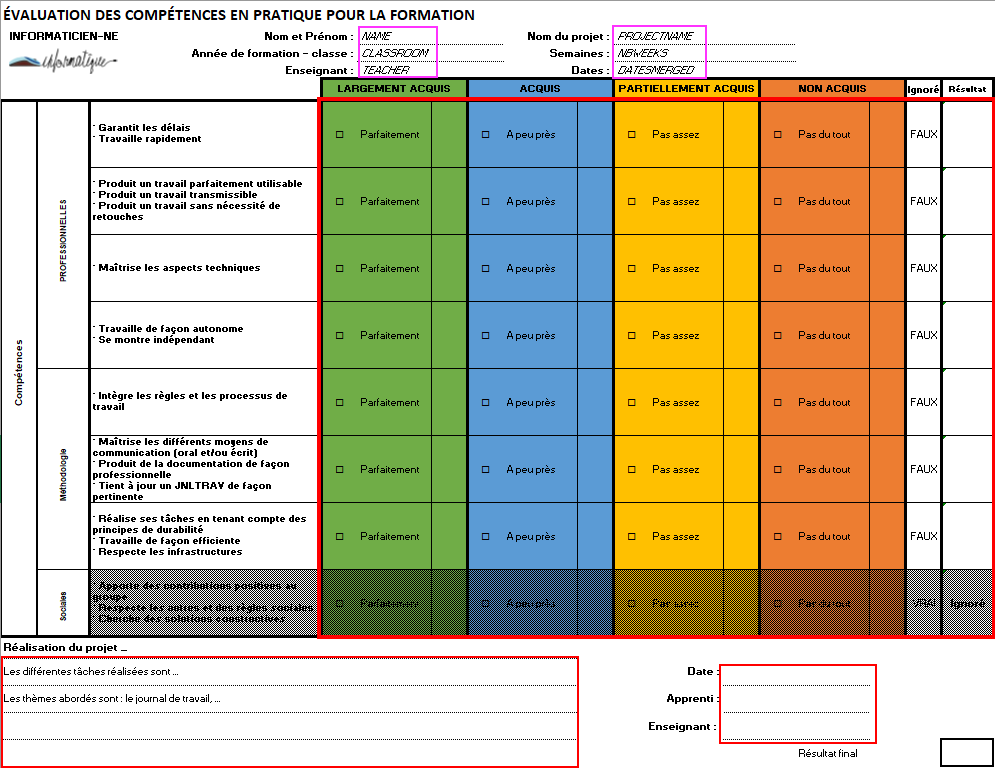


Figure Modèle d'auto-évaluation

**03-synthese-eval.xlsm :**

Ce fichier est le modèle de synthèse des notes. Une fois les auto-évaluations rapatriées, il contient un onglet par évaluation. De plus, dans l’onglet ‘’MASTER‘’, se trouvent trois boutons permettant respectivement de : Rapatrier les notes dans le tableau, vider le tableau et exporter les auto-évaluations en PDF.

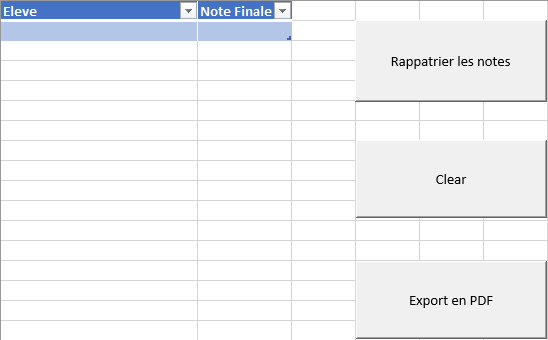


Figure Modèle de synthèse

### Fichiers de scripts

Voici les différents fichiers de script que nous allons aborder :

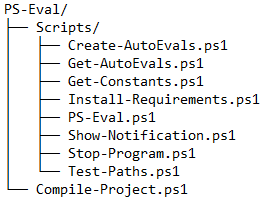


Figure Arborescence du projet

#### Create-AutoEvals.ps1

Ce script permet de créer les auto-évaluations des élèves selon un fichier de configuration et un modèle. Ces auto-évaluations seront créées dans un répertoire. Ces fichiers sont définis en arguments.

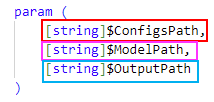


Figure Paramètres pour la création des auto-évaluations

Toutes les constantes nécessaires au script sont récupérées depuis le fichier

‘’Get-Constants.ps1‘’.

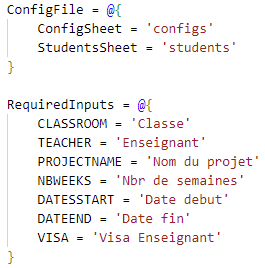
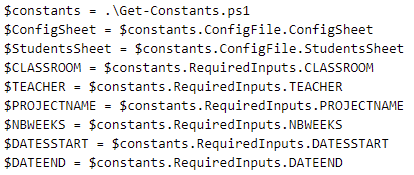


Figure Obtention des constantes

Figure Déclaration des constante

Le script test ensuite ces différents points :

* Vérifie que le dossier de sortie existe et sinon le crée.
* Vérifie que le fichier de configuration et de modèle existe
* Vérifie que le fichier de configuration contient bien tous les champs nécessaires

Les configurations et la liste des élèves sont récupérés à l’aide du module ImportExcel.



Figure Import du fichier de configurations

Une boucle ‘’Foreach‘’ parcourt ensuite la liste des élèves pour créer leurs auto-évaluations.

Le ‘’ComObject‘’ Excel est instancié pour pouvoir interagir avec les fichiers Excel.

Une erreur est générée si Excel n’est pas installé ou configuré.

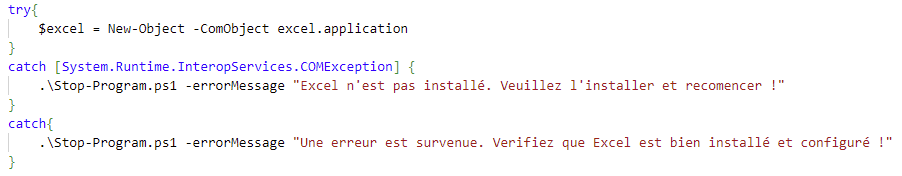


Figure Création de l’Object Excel

Le fichier modèle est importé, puis les cellules de titre sont modifiées avec les valeurs du fichier de configuration.

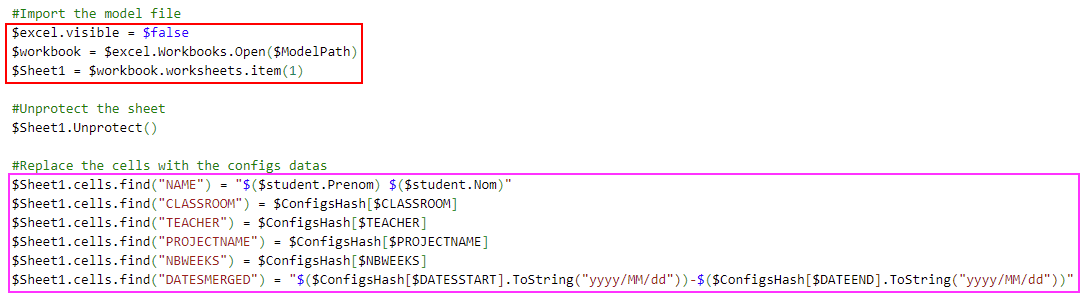


Figure Génération des auto-évaluations

Les fichiers des auto-évaluations sont enregistrés au format ‘’AutoEval-Prenom-Nom.xlsx‘’

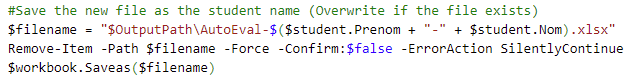


Figure Enregistrement des auto-évaluations

#### Get-AutoEvals.ps1

Ce script rapatrie les auto-évaluations remplies par les élèves, puis crée une synthèse des notes selon le modèle. Ces auto-évaluations sont récupérées d’un répertoire défini en arguments

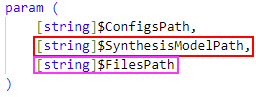


Figure Paramètres pour le rapatriement des auto-évaluations

Toutes les constantes nécessaires au script sont récupérées depuis le fichier

‘’Get-Constants.ps1‘’.

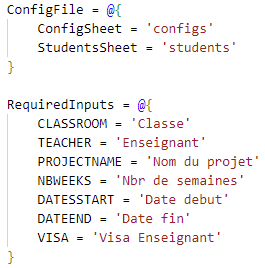
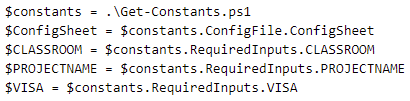


Figure Obtention des constantes

Figure Déclaration des constantes

Le script test ensuite ces différents points :

* Vérifie que le dossier contenant les auto-évaluations existe.
* Vérifie que le fichier de configuration et de modèle existe
* Vérifie que le dossier contenant les auto-évaluations en contient au moins une

Les configurations sont récupérées à l’aide du module ImportExcel.



Figure Importation du fichier de configurations

Une boucle ‘’Foreach‘’ parcourt ensuite les fichiers du dossier contenant les auto-évaluations pour les rapatrier.

Le ‘’ComObject‘’ Excel est instancié pour pouvoir interagir avec les fichiers Excel.

Une erreur est générée si Excel n’est pas installé ou configuré.

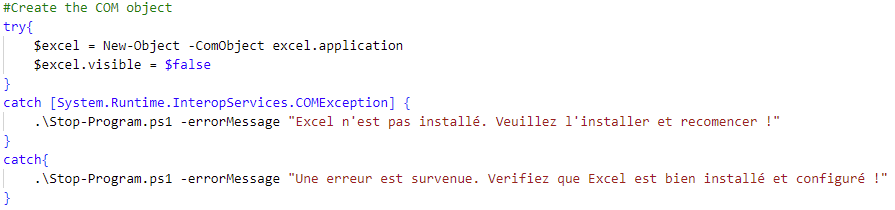


Figure Création de l’Object Excel

Tous les fichiers d’auto-évaluations sont ajoutés au modèle de synthèse en tant qu’onglet Excel.

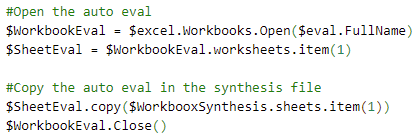


Figure Création de la synthèse

Une fois cette opération faite pour tous les fichiers, le modèle est SaveAs en tant que fichier ‘’xlsm‘’. C’est un fichier Excel prenant en charge les macros. Le nom du fichier est le suivant : ‘’AutoEvals-NomProjet-Classe-Prof-1.xlsm‘’

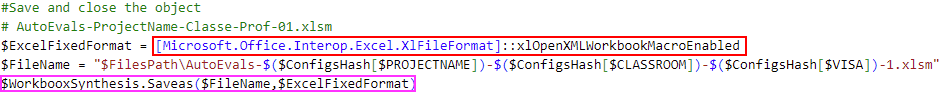


Figure Enregistrement de la synthèse en xlsm

#### Get-Constants.ps1

Ce script contient toutes les constantes nécessaires au fonctionnement du projet.

Il retourne un Object PowerShell qui sera accédé par les scripts l’appelant.

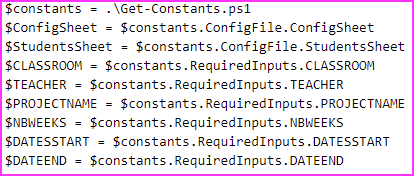
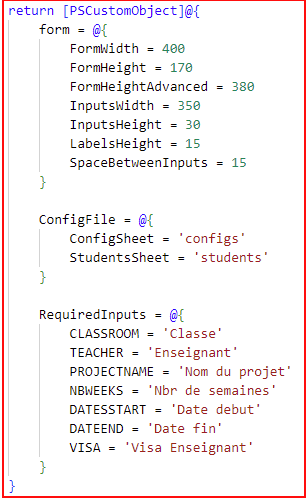


Figure Déclaration des constantes

Figure Obtention des constantes

#### Install-Requirements.ps1

Ce script permet d’automatiser l’installation des modules et autres installations requises. Un test est effectué pour voir si le module est installé et si ce n’est pas le cas, on l’installe. Pareil pour le répertoire ‘’PSGallery‘’.

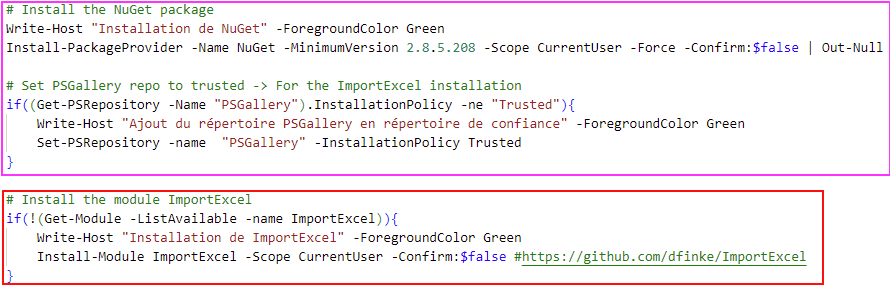


Figure Installation des modules

#### PS-Eval.ps1

Ce script est la base du projet. Il affiche une interface graphique en Windows Forms afin de permettre à l’utilisateur de créer et rapatrier les auto-évaluations.

Toutes les constantes nécessaires au script sont récupérées depuis le fichier

‘’Get-Constants.ps1‘’.

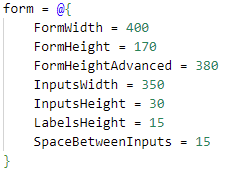
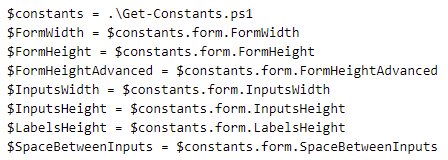


Figure Obtention des constantes

Figure Déclaration des constantes

La console PowerShell est ensuite masquée, pour avoir uniquement le formulaire affiché à l’utilisateur

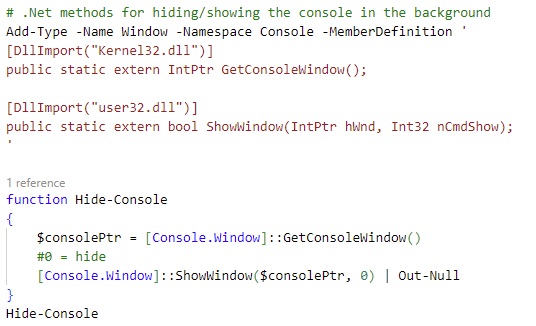


Figure Masquer la console PowerShell

Le formulaire est créé en Windows Forms

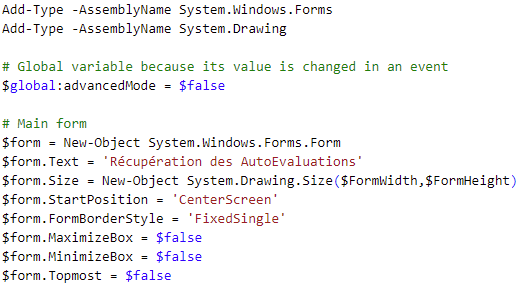


Figure Création du formulaire

Sur les boutons de Création et de Rapatriement, un évènement ‘’Add\_Click‘’ est ajouté. Cet évènement s’exécute lors d’un clic sur le bouton.

Ensuite, le script de création ou de rapatriement est lancé. Si une erreur est retournée, on l’affiche avec une MessageBox

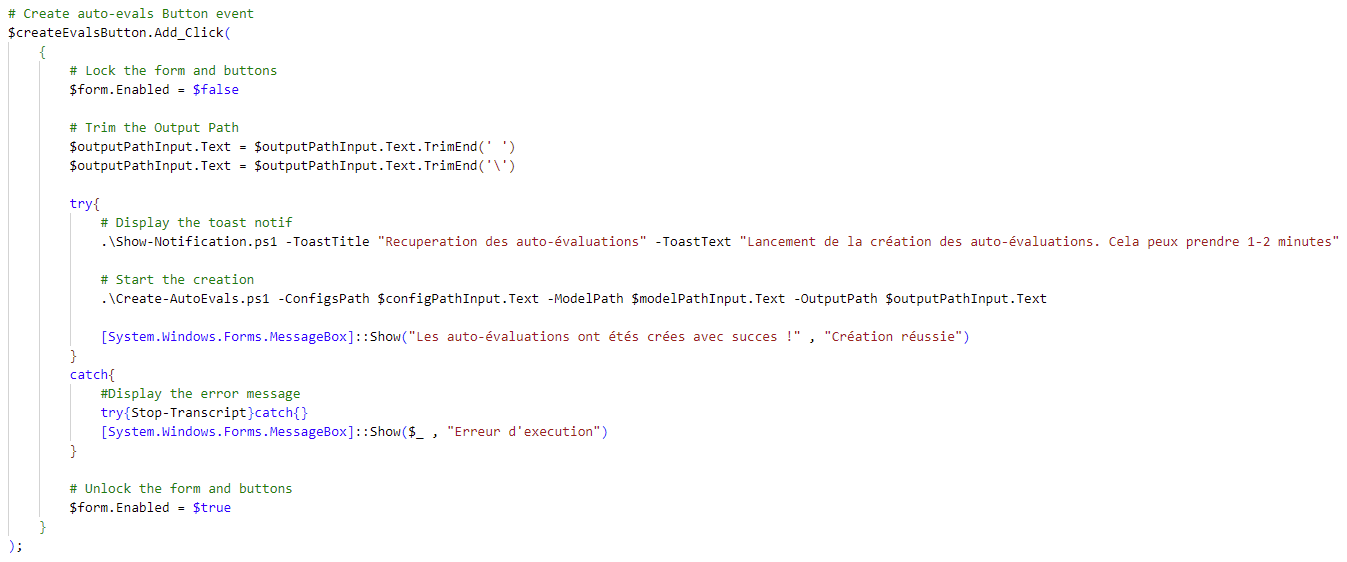


Figure Démarrage de la création des auto-évaluations

#### Show-Notification.ps1

Ce script affiche une notification Windows avec le titre et le contenu spécifié en paramètres.

Une image contenant texte

Description générée automatiquement

Figure Notification

Figure Titre et contenu de la notification

Les notifications sont définies en XML, Sous le format suivant :

Une image contenant texte

Description générée automatiquement

Figure Code XML de la notification

Il existe bien un module PowerShell prenant en charge l’affichage de notifications.

Cependant, le module doit être installé ce qui prend du temps. Le but principal de ce script, est d’afficher une notification pour avertir l’utilisateur que l’exécution du script peux prendre du temps. Donc si la notification d’avertissement n’arrive pas tout de suite cela n’a pas vraiment d’utilité. C’est pour cette raison que nous n’avons pas utilisé le module, mais recodé en XML.

#### Stop-Program.ps1

Ce script permet d’arrêter l’exécution des scripts, tout en stoppant la journalisation des actions et en retournant un message d’erreur spécifié en paramètres.

Une image contenant texte

Description générée automatiquement

Figure Fonction pour arrêter l'exécution

#### Test-Paths.ps1

Ce script teste si plusieurs chemins de fichiers / dossiers donnés en arguments existent. Retourne une liste avec tous les fichiers / dossiers non existants. Cette liste est vide si tout les fichiers / dossiers existent.

Une image contenant texte

Description générée automatiquement

Figure Fonction pour tester l'existence de chemins

#### Compile-Project.ps1

Ce script Récupère le contenu de tous les scripts mentionnés ci-dessus, et les compile en un seul. Cela permet d’utiliser le projet avec plus de facilitée.

Les paramètres sont définis comme ci-dessous. Ce sont les valeurs proposées au prochain développeur.

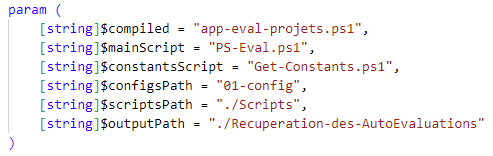


Figure Paramètres pour la compilation

Tous les scripts sont récupérés, puis un dossier de sortie est créé et est défini comme répertoire courant. Les scripts ‘’PS-Eval.ps1‘’ et ‘’Get-Constants.ps1‘’ sont exclus. Vous verrez la raison à l’étape suivante.

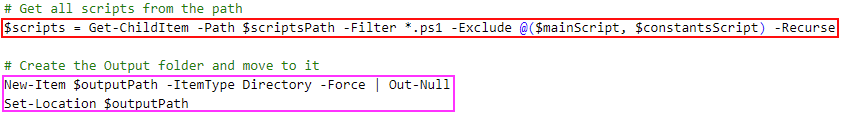


Figure Récupération des scripts

Tout le contenu des scripts est récupéré, puis les ‘’.ps1‘’ et ‘’.\‘’ sont retirés. Car la syntaxe pour appeler les scripts est la suivante : ‘’.\Script-Exemple.ps1‘’ et quand les fonctions sont définies dans le même script la syntaxe est la suivante : ‘’Script-Exemple‘’. Le fichier ‘’Get-Constants.ps1‘’ doit se retrouver en haut du script, car les autres scripts l’appellent tous. Le script ‘’PS-Eval.ps1‘’ doit lui être en dernier car c’est lui qui lance les autres scripts.

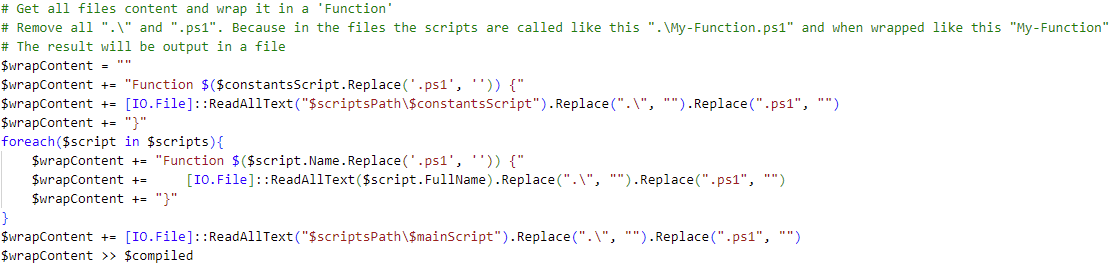


Figure Compilation

Le dossier ‘’01-config‘’ est copié dans le dossier de sortie.



Figure Copie du dossier de configurations

Un fichier .bat est créé pour permettre à l’utilisateur de lancer facilement le projet et pour contourner la police d’exécution ‘’Restricted‘’.



Figure Création du fichier de lancement

### Librairies externes

La seule librairie externe utilisée dans ce projet est ImportExcel.

C’est une libraire qui permet d’effectuer des opérations Excel basiques très simplement. Elle à été utilisée dans les scripts ‘’Create-AutoEvals.ps1‘’ et ‘’Get-AutoEvals.ps1‘’

## Description des tests effectués

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Impact en cas d’échec | Résultat attendu | Résultat obtenu | Conclusion | Temps requis pour corriger et / ou finaliser |
| Le script démarre et affiche l’interface | Impossibilité d’utiliser le projet. | L’interface graphique est affichée | OK 100% | Rien à signaler |  |
| Les auto-évaluations sont créées avec les données spécifiées dans le fichier de configuration | Impossibilité d’utiliser le projet. | Les auto-évaluations sont correctement créées | OK 100% | Rien à signaler |  |
| Les résultats des auto-évaluations sont correctement récupérés | Synthèse des notes impossible | Les résultats sont rapatriés | OK 100% | Rien à signaler |  |
| Les commentaires des élèves sont correctement récupérés | Correction par l’enseignant biaisée | Les commentaires sont rapatriés | OK 100% | Rien à signaler |  |
| Les auto-évaluations sont correctement exportées en PDF | Envoie des notes corrigées doit être fait à la main | Les auto-évaluations sont exportées en PDF | OK 100% | Rien à signaler |  |
| Aucun droit ni configuration nécessitant un spécialiste n’est nécessaire | Utilisation limitée selon les personnes | Tout utilisateur peux utiliser le projet sans erreurs | OK 100% | Rien à signaler |  |
| Fonctionnement sur Windows 11 | Le projet va très vite être obsolète |  | OK 100% | Le script sera utilisable lors de la migration du parc vers Windows 11 |  |

## Liste des documents fournis

- Le rapport de projet

- Le code source du projet

- Manuel d’utilisation

- Manuel de modifications

- Journal de travail

- Planification détaillée

Les documents cités ci-dessus sont disponible dans la partie annexe de ce document.

## Glossaire

1. Foreach : Boucle permettant notamment de parcourir un tableau
2. ComObject : Objet PowerShell permettant de manipuler des composants Microsoft
3. PSGallery : Référentiel de modules PowerShell
4. MessageBox : Boite affichant un message a l’utilisateur
5. ExecutionPolicy : Police d’exécution de scripts PowerShell
6. Notification Toast : Notification Windows

# Conclusion

Pour conclure ce projet, je vais procéder en plusieurs étapes. Les étapes sont les objectifs, les difficultés particulières, les suite possible pour le projet, un bilan personnel et pour finir un bilan de planification.

**Les objectifs :**

Nous allons aborder les points du cahier des charger et voir s’ils ont été atteints.

* **Les valeurs proposées par les élèves sont récupérées correctement :**

Ce point est à 100% fonctionnel

* **Les commentaires des élèves sont récupérés correctement :**

Ce point est à 100% fonctionnel

* **Toutes les valeurs des indicateurs sont récupérées correctement :**

Ce point est à 100% fonctionnel

* **Le dispositif est pertinent et simple à utiliser :**

Ce point est compliqué à valider car tout le monde est différent et les gens ne trouvent pas tous les mêmes choses faciles. Cependant en vue des tests les retours concernant ce point étaient très positifs.

* **Le dispositif est fiable. En cas d’incidents, les messages d’erreurs sont pertinents :**

Il est très difficile de dire à 100% que le dispositif est fiable car il n’a pas encore été testé à grande échelle. Cependant avec les différents tests effectués, ce point est considéré comme réalisé.

* **Une modification de la structure de la grille est possible :**

Ce point est à 100% fonctionnel

* **Fonctionne sans autorisation spéciale, ni configuration exigeant l’intervention d’un spécialiste système :**

Ce point est à 100% fonctionnel

**Les difficultés particulières :**

Une des grosses difficultés auxquelles nous nous sommes confrontés, à été la création des auto-évaluations. Car avec PowerShell ce n’est pas très simple d’interagir avec les fichiers Excel. De plus il y’a une quantité limitée de documentation à ce sujet. Cela nous à donc bien challengé lors de l’apprentissage de la manière de faire.

**Les suites possibles du projet :**

Le projet va être mis en production auprès des enseignants volontaires. Ils l’utiliseront à leurs guises et pourront le modifier / tenir à jours comme ils le désirent. Le code source étant accessible sur GitHub ils pourront sans soucis effectuer les modifications souhaitées.

**Bilan personnel :**

Nous avons particulièrement apprécié ce projet dans l’ensemble. Aimant beaucoup automatiser les actions, ce projet était très agréable. De plus nous avons pu consolider nos connaissances dans ce domaine. Nous avons également aimé apprendre à gérer des fichiers Excel avec PowerShell, ce qui peux être très utile dans le monde du travail. Ce sont donc des connaissances qui nous seront utile tout au long de notre carrière.

**Bilan de planification :**

Concernant notre planification, vous pouvez voir sur le graphique ci-dessous les principales différences entre la **planification** et la **réalisation**.

Premièrement, nous pouvons constater 90 quarts d’heures en imprévus. Ce qui correspond à une semaine planifiée que nous avons finalement eue en moins pour la résiliation de notre projet. Nous pouvons voir que dans le point ‘’Informer‘’ qui correspond notamment au rapport, nous à pris plus de temps que prévu. Les tests ont cependant été très performants et concluant. En revanche le point ‘’Autre’’ qui correspondait surtout à la préparation de la présentation à été beaucoup moins réalisé, à cause de la semaine en moins.

# Annexes

## Résumé du rapport du TPI / version succincte de la documentation

Situation de départ :

Le but de ce projet était de proposer un système d’automatisation pour la gestion des auto-évaluations. Il existait à la base un fichier modèle, et les auto-évaluations étaient créés à partir de ce fichier. Tout était fait à la main par les enseignants, ce qui était très couteux en temps.

Mise en œuvre :

Le programme devait permettre d’automatiser la création des auto-évaluations selon des données spécifiées (Nom du projet, durée, liste des élèves…). De plus il était aussi demandé de pouvoir créer une synthèse avec toutes les auto-évaluations une fois remplies par les élèves. Cette synthèse permet de corriger les auto-évaluations, puis une fonctionnalité devait permettre de les exporter en PDF.

Résultats :

Un script permettant la création et le rapatriement des auto-évaluations. Un fichier pour les configurations (Nom du projet, durée, liste des élèves…), un fichier modèle d’auto-évaluation et un fichier modèle de synthèse.

## Sources – Bibliographie

1 : GitHub du module PowerShell ImportExcel

<https://github.com/dfinke/ImportExcel>

2 : Utilisation de Excel avec PowerShell

<https://techexpert.tips/powershell/powershell-creating-excel-file/>

3 : Ouverture d’un fichier Excel à l’aide de PowerShell

<https://stackoverflow.com/questions/37665118/how-to-open-excel-workbook-from-powershell-for-automation>

4 : Formatage de date en PowerShell

<https://stackoverflow.com/questions/2249619/how-to-format-a-datetime-in-powershell>

5 : Utilisation de JSON avec PowerShell

<http://ramblingcookiemonster.github.io/PowerShell-Configuration-Data/>

6 : Création d’un module manifeste

<https://learn.microsoft.com/en-us/powershell/scripting/developer/module/how-to-write-a-powershell-module-manifest?view=powershell-7.3>

7 : Vider un COM Object en PowerShell

<https://social.technet.microsoft.com/Forums/office/en-US/e5d8594b-b14b-4a54-913c-61089b5d9ab4/release-or-delete-a-com-object-from-powershell?forum=winserverpowershell>

8 : Transcription des logs

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.host/start-transcript?view=powershell-7.3>

9 : Gérer les exceptions en PowerShell

<https://learn-powershell.net/2015/04/09/quick-hits-finding-exception-types-with-powershell/>

10 : Problèmes avec un ‘’Get-ChlidItems‘’

<https://stackoverflow.com/questions/41700636/get-childitem-doesnt-work-with-include>

11 : Copier une feuille Excel dans une autre en PowerShell

<https://stackoverflow.com/questions/3226096/copy-excel-worksheet-from-one-workbook-to-another-with-powershell>

12 : Macro VBA pour modifier un tableau Excel

<https://learn.microsoft.com/en-us/answers/questions/864246/powershell-vba-and-excel>

13 : Parcourir toutes les feuilles Excel d’un fichier en VBA

<https://support.microsoft.com/en-us/topic/macro-to-loop-through-all-worksheets-in-a-workbook-feef14e3-97cf-00e2-538b-5da40186e2b0>

14 : Vider un tableau Excel en VBA

<https://vbaf1.com/table/clear-table-content/>

15 : Déclaration de variable en VBA

<https://www.automateexcel.com/vba/string-data-type/>

16 : Ajouter des données à un tableau Excel en VBA

<https://officetuts.net/excel/vba/add-data-to-table-using-vba/>

17 : Méthode de remplacement de caractère dans une string en VBA

<https://wellsr.com/vba/2016/excel/vba-replace-function-to-replace-characters-in-string/>

18 : Création de formulaire Windows Forms en PowerShell

<https://learn.microsoft.com/en-us/powershell/scripting/samples/creating-a-custom-input-box?view=powershell-7.3>

19 : Argument avec multiples valeurs en PowerShell

<https://stackoverflow.com/questions/15120597/passing-multiple-values-to-a-single-powershell-script-parameter>

20 : Evènement sur click de bouton en PowerShell

<https://social.technet.microsoft.com/wiki/contents/articles/25911.how-to-add-a-powershell-gui-event-handler-part-1.aspx>

21 : Création d’un label cliquable en PowerShell

<https://social.technet.microsoft.com/wiki/contents/articles/28746.powershell-gui-how-to-create-a-link-label-using-winforms.aspx>

22 : Stocker le contenu d’un fichier dans une variable

<https://stackoverflow.com/questions/7976646/powershell-store-entire-text-file-contents-in-variable>

23 : Création d’une variable globale en PowerShell

<https://stackoverflow.com/questions/68084565/powershell-wpf-boolean-resets-its-value-after-function>

24 : Création d’objets customisés en PowerShell

<https://powershellbyexample.dev/post/custom-object/>

25 : Affichage de notifications toast en PowerShell

<https://www.starwindsoftware.com/blog/how-to-display-notifications-on-windows-10-using-powershell>

26 : Affichage de notifications toast en PowerShell

<https://den.dev/blog/powershell-windows-notification/>

## Journal de travail

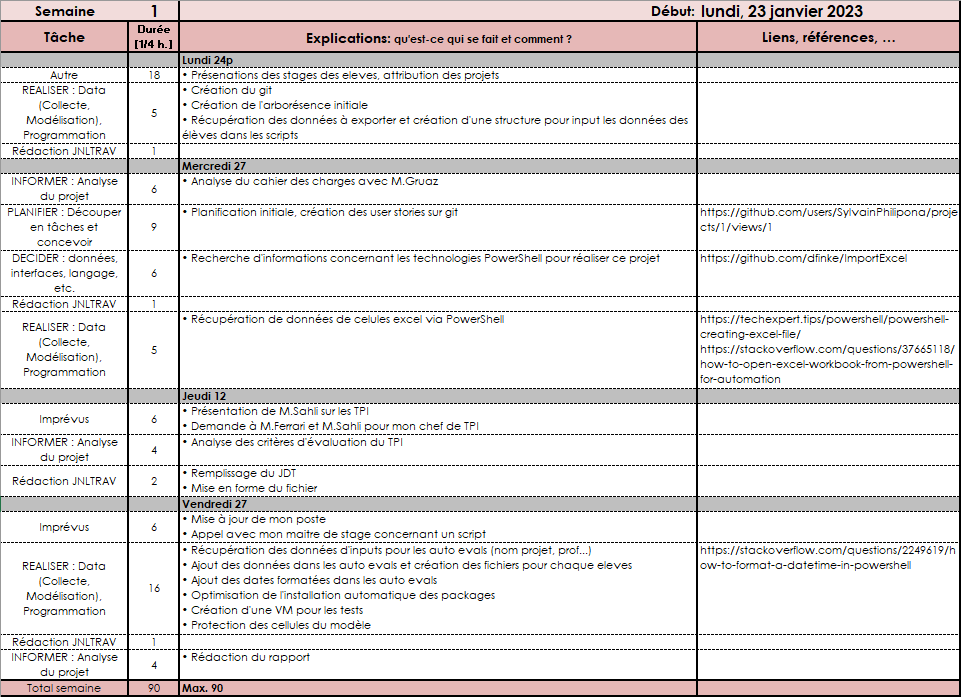


Figure Journal de travail semaine 1

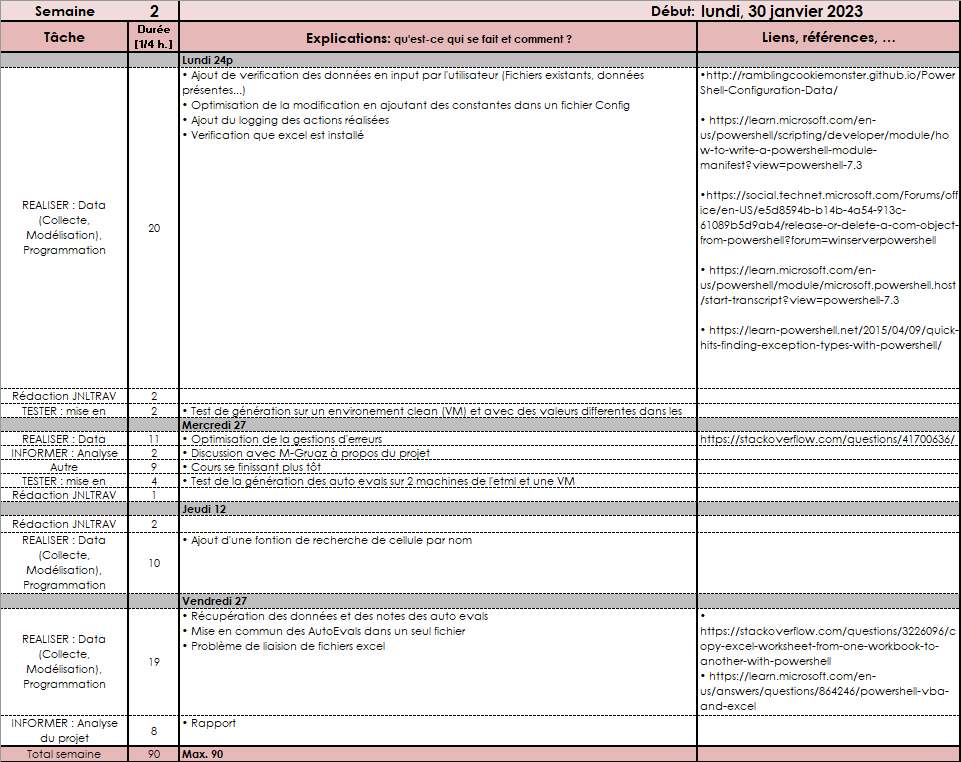


Figure Journal de travail semaine 2

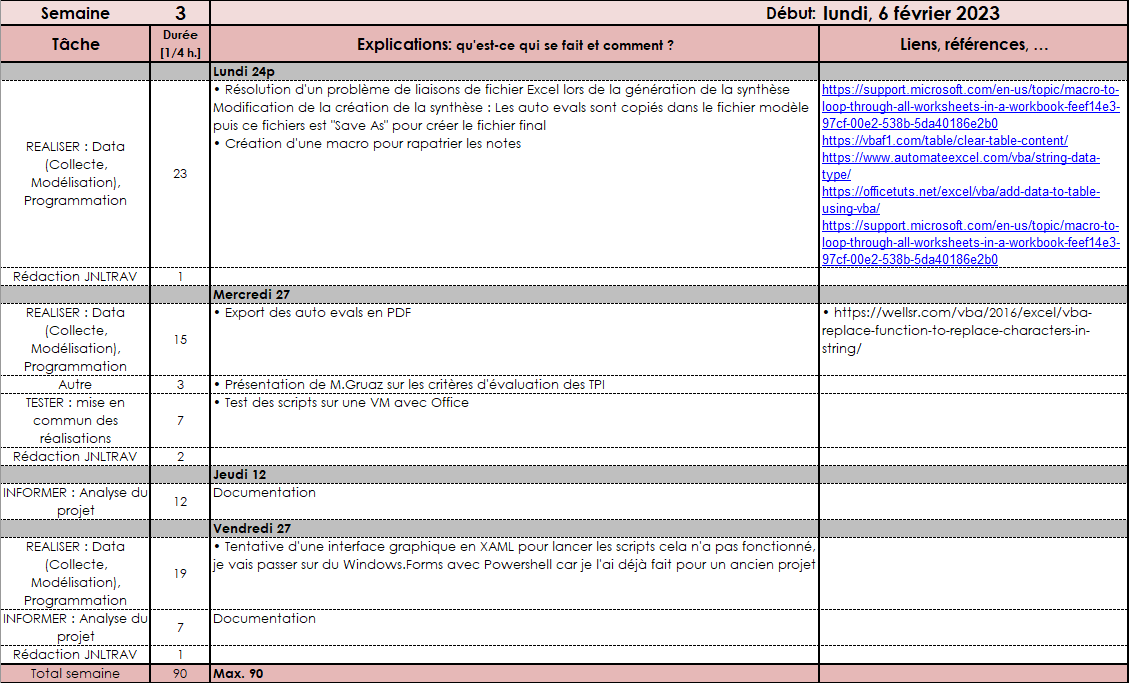


Figure Journal de travail semaine 3

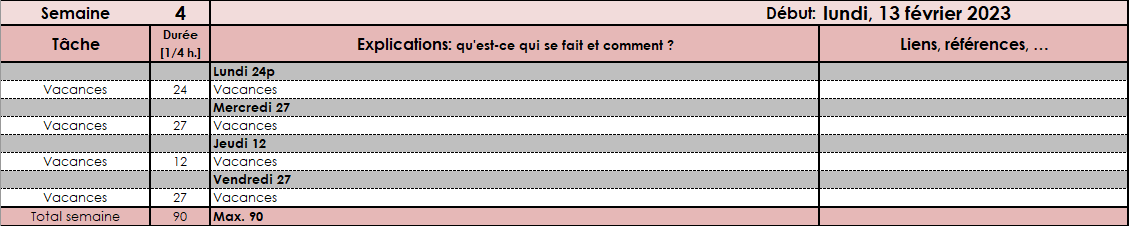


Figure Journal de travail semaine 4

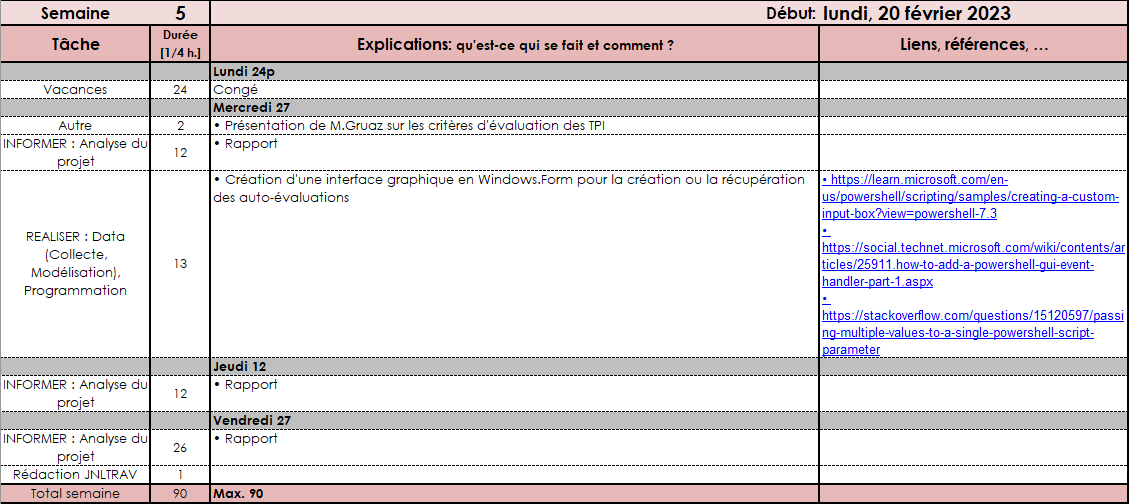


Figure Journal de travail semaine 5

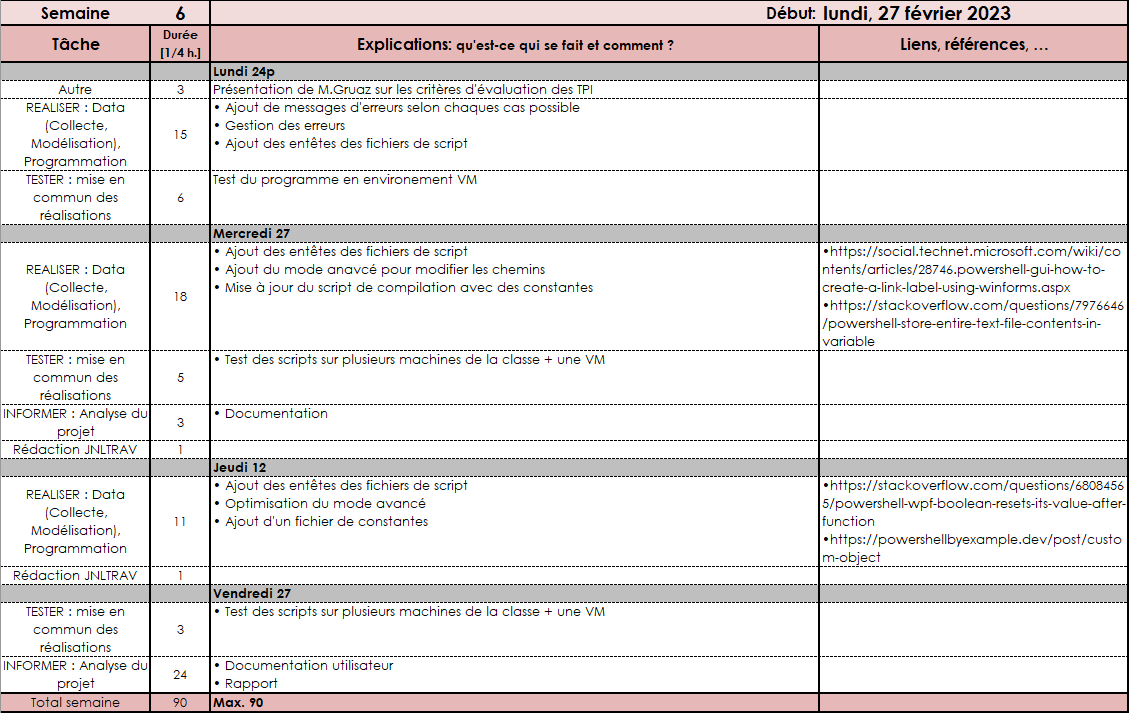


Figure Journal de travail semaine 6

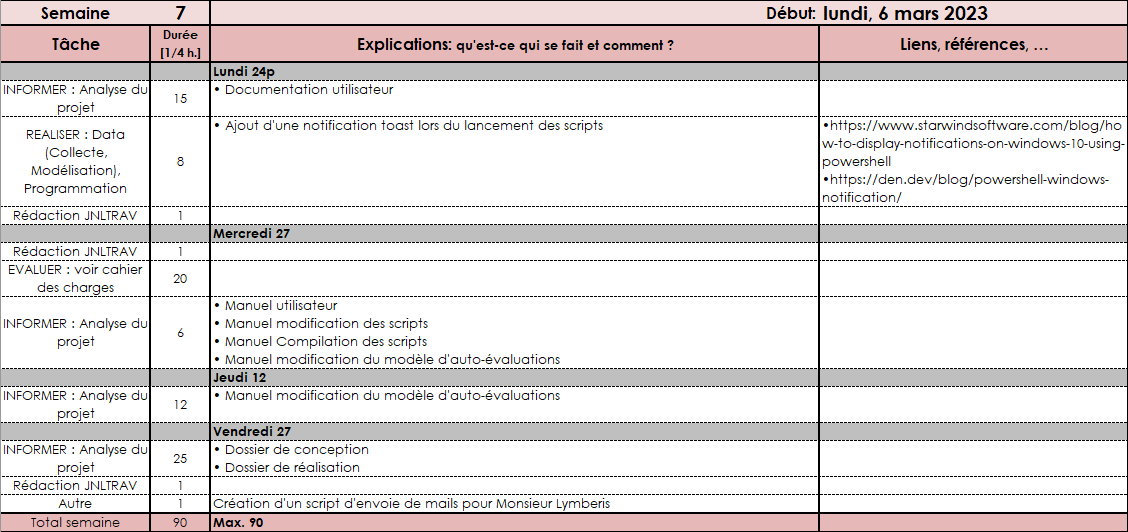


Figure Journal de travail semaine 7

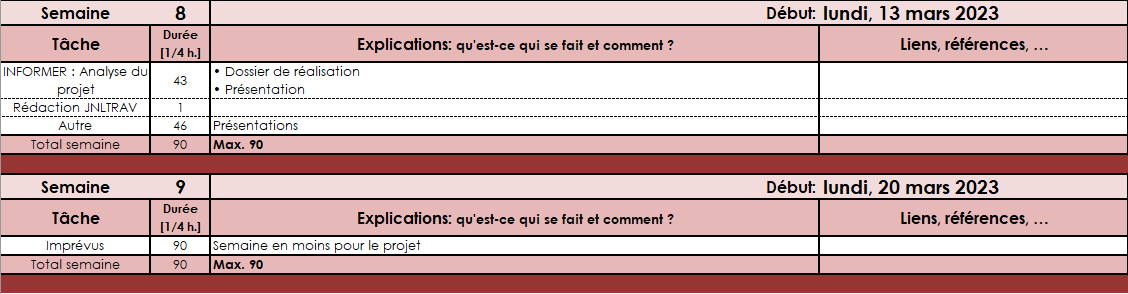


Figure Journal de travail semaine 8

## Planification détaillée

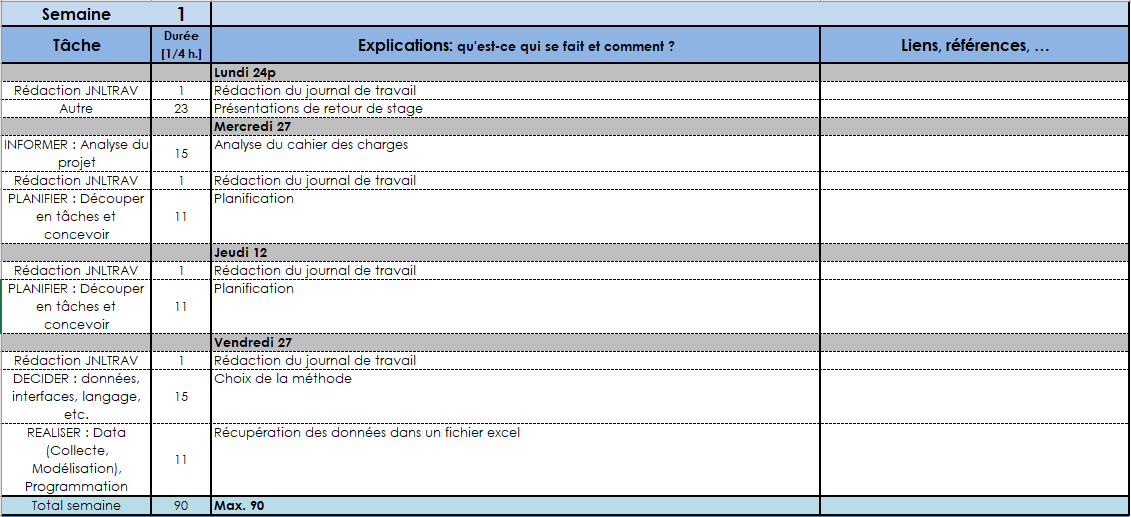


Figure Planification semaine 1

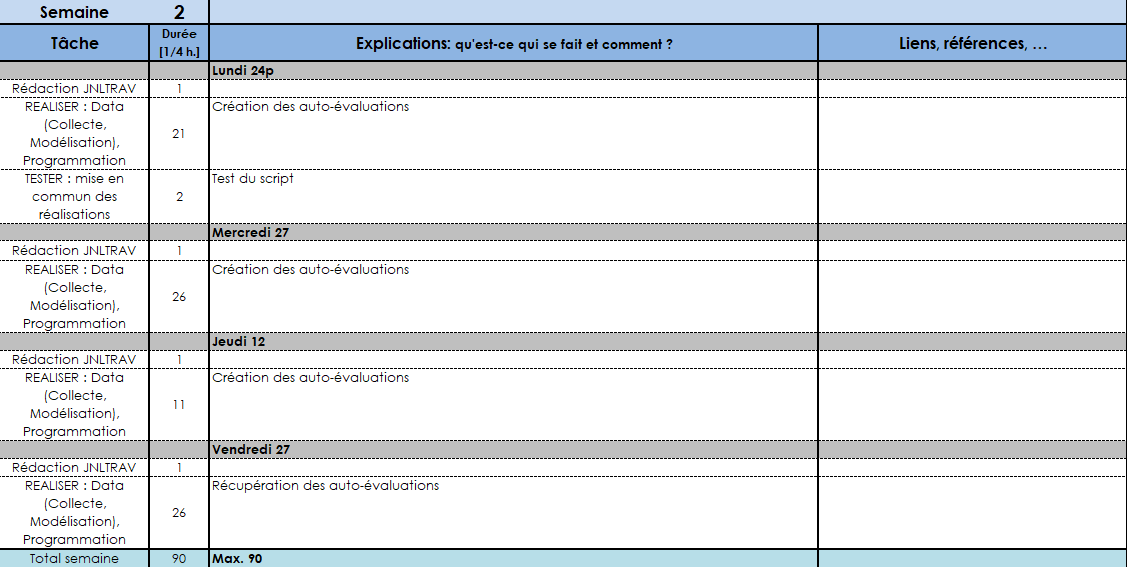


Figure Planification semaine 2

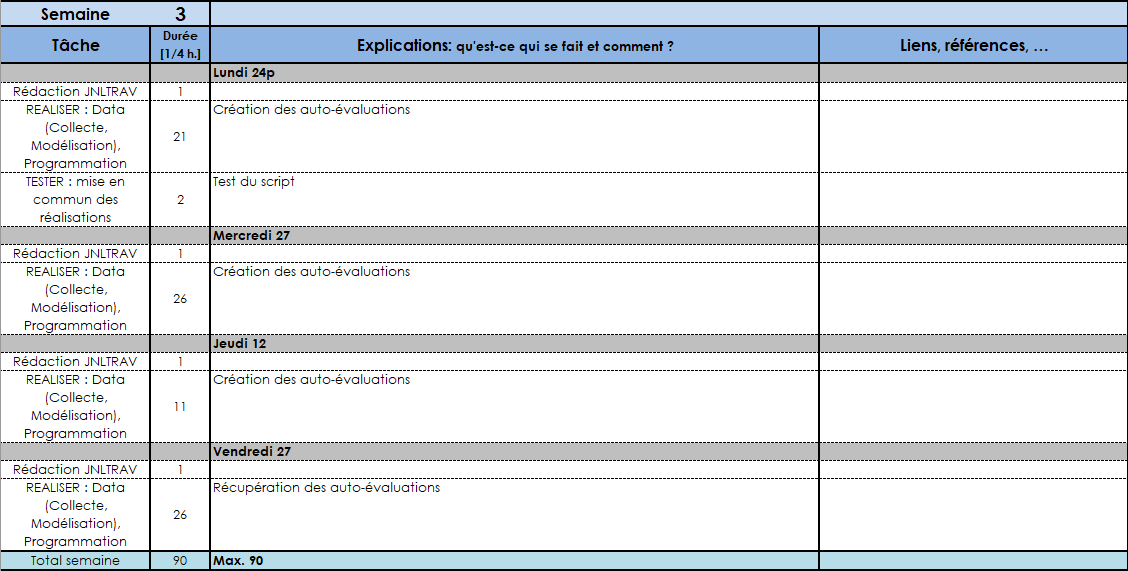


Figure Planification semaine 3

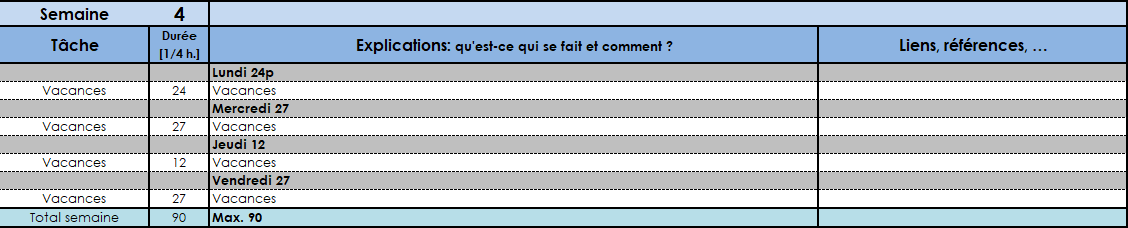


Figure Planification semaine 4

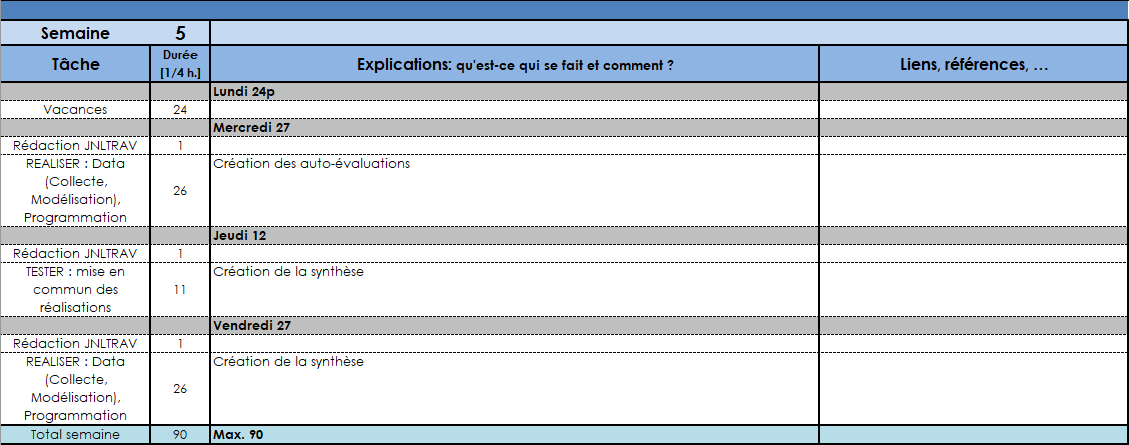


Figure Planification semaine 5

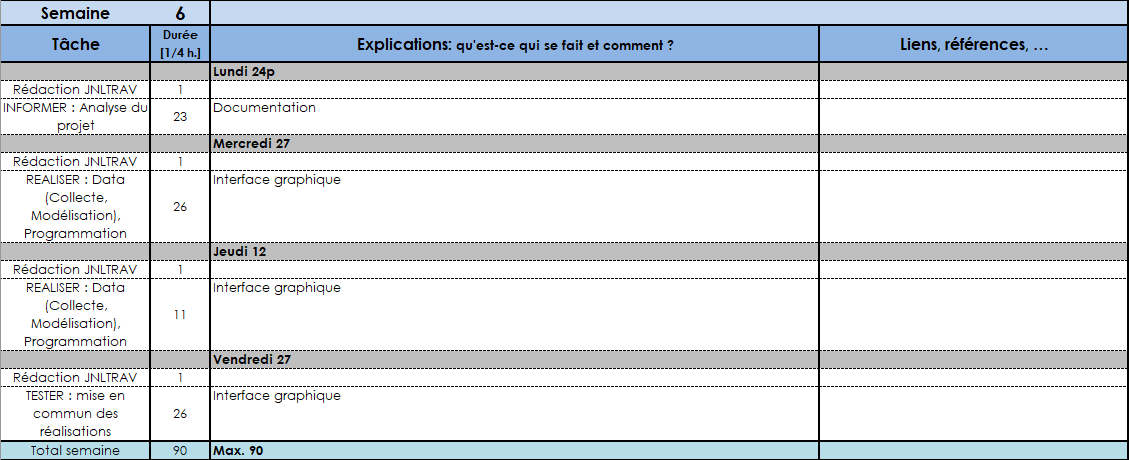


Figure Planification semaine 6

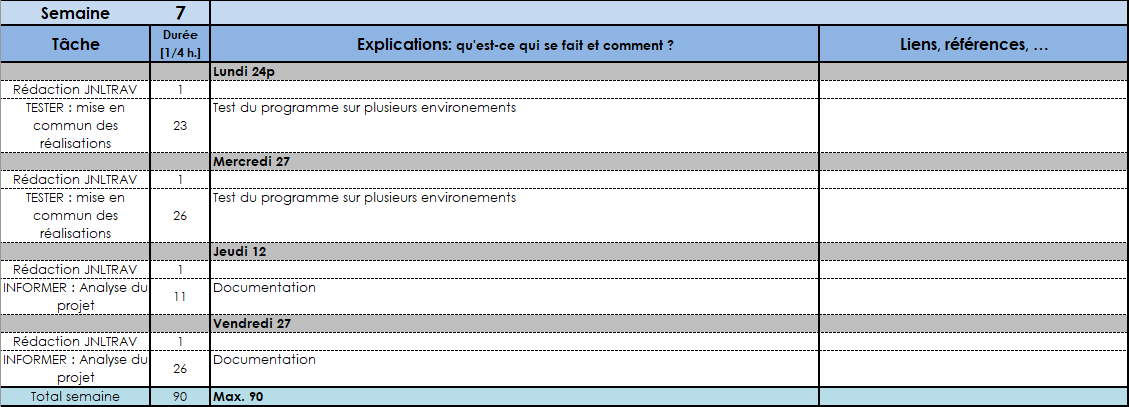


Figure Planification semaine 7

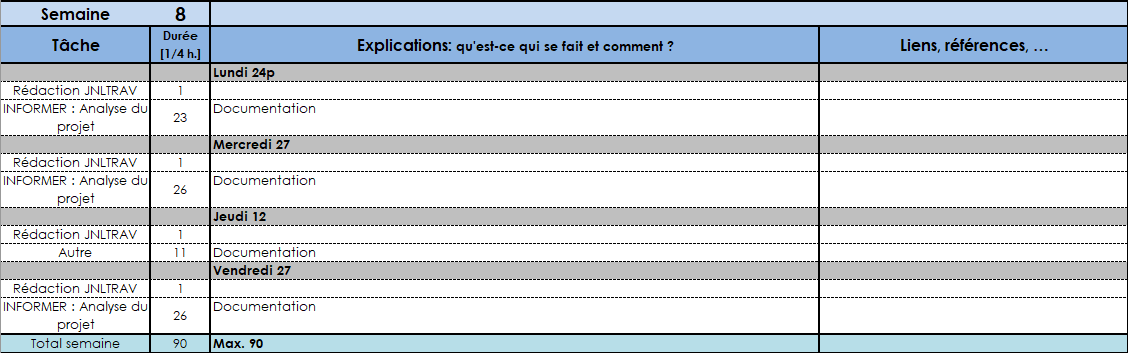


Figure Planification semaine 8

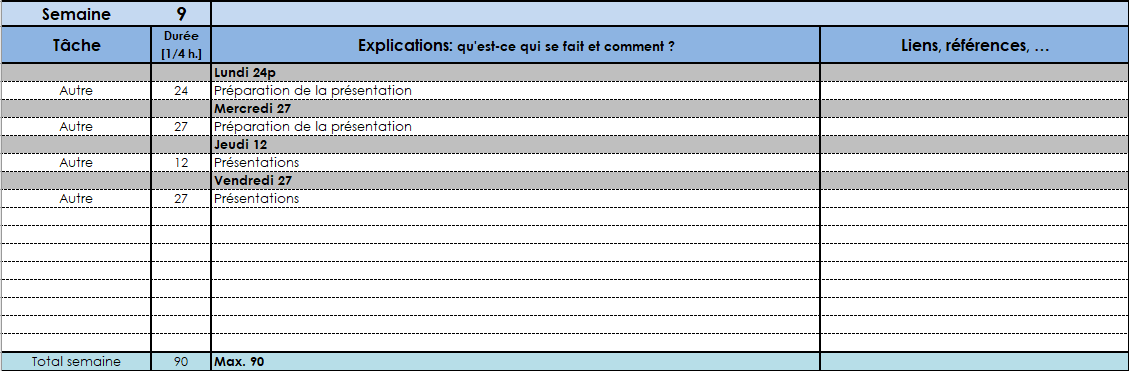


Figure Planification semaine 9

## Manuel de modifications

### Modification des scripts

Tous les fichiers de scripts se trouvent dans le dossier ‘’Scripts’’ du GitHub.

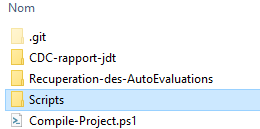


Figure Dossier de scripts

Le plus simple est d’ouvrir le dossier depuis Visual Studio Code

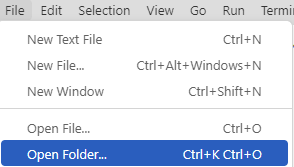
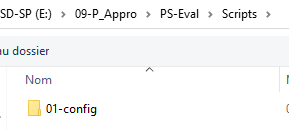
 🡪 

Figure Ouverture du dossier dans VS Code

Figure Dossier de config

Vous trouverez sur le côté gauche la liste des fichiers de scripts

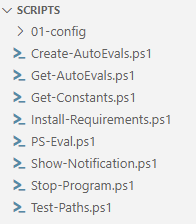


Figure Liste des scripts

⚠ Pour modifier les scripts il faut suivre quatre points :

* Tous les scripts doivent être dans le dossier ‘’Scripts’’
* Le script ‘’PS-Eval.ps1’’ est le script principal qui lance les autres
* Quand vous appelez une fonction, par exemple ‘’Show-Notification.ps1’’ veillez bien à utiliser cette syntaxe : ‘’.\Show-Notification.ps1’’
* Quand vous utilisez un chemin relatif veillez bien à utiliser cette syntaxe : ‘’./Chemin/etc’’

Si un de ces points n’est pas respecté, la compilation des scripts ne fonctionnera pas !

### Compilation des scripts

Dans le dossier du GitHub, vous trouverez le fichier ‘’Compile-Project.ps1’’

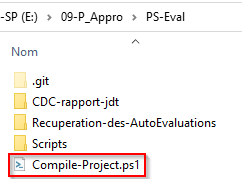


Figure Fichier de configurations

Faites un clic-droit sur ce fichier, puis ‘’Exécuter avec PowerShell’’

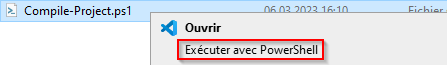


Figure Compilation

Cela va créer / remplacer le dossier ‘’Recuperation-des-AutoEvaluations’’ et compiler tous les scripts en un seul. Un fichier ‘’start.bat’’ va être crée dans le but d’ignorer la ‘’ExecutionPolicy’’. Le dossier ‘’01-config’’ contenant les fichiers de configurations va être copié.

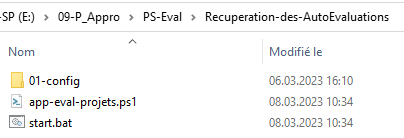


Figure Résultat de la compilation

⚠ Il est nécessaire d’avoir respecté les points mentionnés dans le chapitre ‘’Modifications des scripts’’

* *Tous les scripts doivent être dans le dossier ‘’Scripts’’*
* *Le script ‘’PS-Eval.ps1’’ est le script principal qui lance les autres*
* *Quand vous appelez une fonction, par exemple ‘’Show-Notification.ps1’’ veillez bien à utiliser cette syntaxe : ‘’.\Show-Notification.ps1’’*
* *Quand vous utilisez un chemin relatif veillez bien à utiliser cette syntaxe : ‘’./Chemin/etc’’*

Si un de ces points n’est pas respecté, le script ne fonctionnera pas !

### Modifications du modèle d’auto-évaluations

#### Préparation à la modification

Dans le dossier ‘’Scripts/01-config’’, ouvrez le fichier ‘’02-modele-grille.xlsx’’

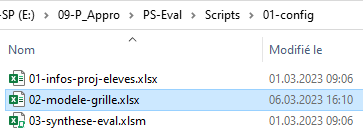


Figure Fichier modèle

La première étape avant d’effectuer des modifications, est d’ôter la protection de la feuille.

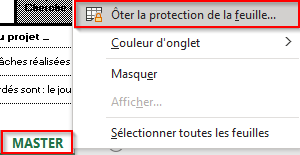


Figure Ôter la protection de la feuille

#### Ajout d’un champ de compétence :

Sélectionnez une ligne des critères d’évaluation, puis faite un clic-droit sur le numéro de ligne. Cliquez ensuite sur ‘’Insérer’’

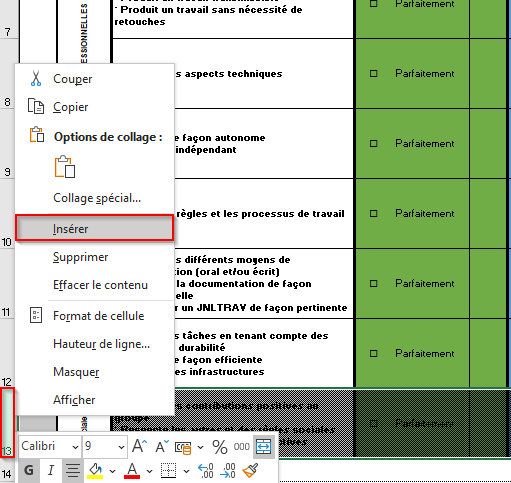


Figure Insertion d'une ligne

Sélectionnez les cellules suivantes de la ligne du dessus, puis double-cliquez sur le carré en bas à droite pour étendre les valeurs et formules.



Figure Duplication des données

Ajoutez ensuite le titre de la compétence et la description

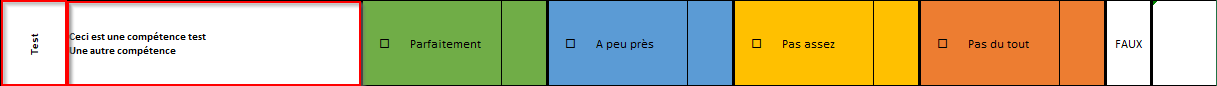


Figure Titre et description

#### Modification des champs de titre :

Les champs de titre sont les cellules contenant les variables qui seront modifiés à la génération des auto-évaluations.



Figure Données de titre

Sur les différentes cellules de titre, définissez une valeur dans la cellule (1), puis nommez cette cellule avec la même valeur (2).

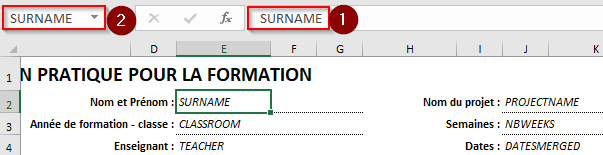


Figure Modification des données de titre

Dans le gestionnaire de noms, supprimez l’ancienne valeur

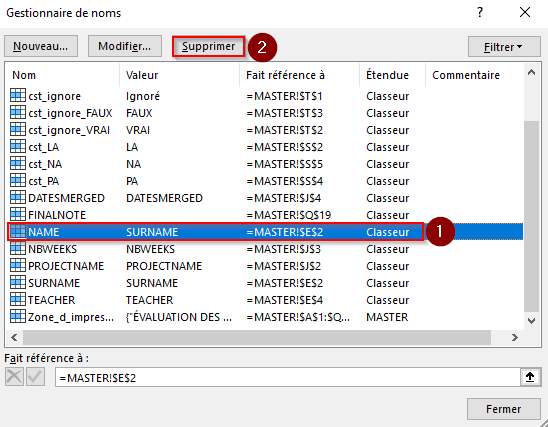


Figure Modification dans le gestionnaire de noms

Une fois ces modifications effectuées, enregistrez et fermez le fichier.

Ouvrez maintenant le fichier ‘’03-synthese-eval.xlsm‘’

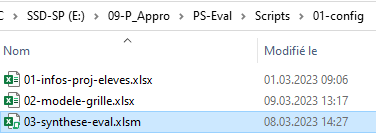


Figure Modèle de synthèse

Dans l’onglet ‘’Développeur‘’ cliquez sur ‘’Visual Basic‘’

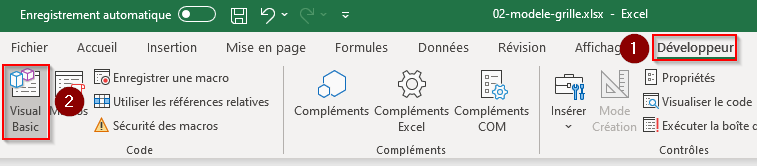


Figure Modification du VBA

Ouvrez l’onglet ‘’VBAProject‘’ puis le module ‘’ModSynthesis‘’.

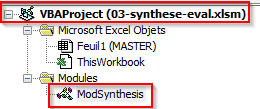


Figure Ouverture du module VBA

Dans la méthode ‘’Create‘’ changez les valeurs suivantes selon votre modification.

Enregistrez et fermez ensuite le fichier.

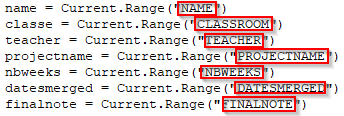


Figure Modification des données de titre dans le VBA

Rendez-vous ensuite dans le fichier ‘’Create-AutoEvals.ps1‘’ du dossier ‘’Scripts‘’

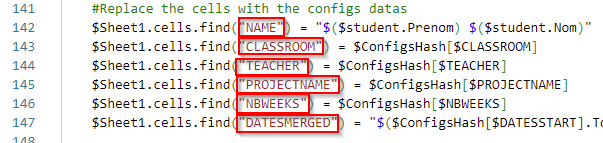


Figure Modification des données de titre dans le PowerShell

#### Protection de la feuille

Une fois toutes les modifications effectuées n’oubliez pas de reprotéger la feuille Excel pour éviter les modifications non voulues.

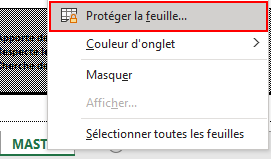


Figure Protéger la feuille

Sélectionnez les checkbox suivantes, puis cliquez sur ‘’OK‘’

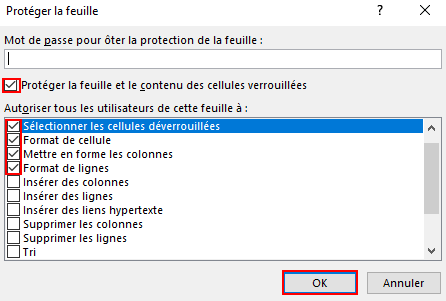


Figure Sélection des options de verrouillage

## Manuel d'Utilisation

### Téléchargement

Pour télécharger le programme vous le trouverez sur [GitHub](https://github.com/SylvainPhilipona/Recuperation-des-AutoEvaluations). Cliquez sur ‘’Code’’ puis ‘’Download ZIP’’

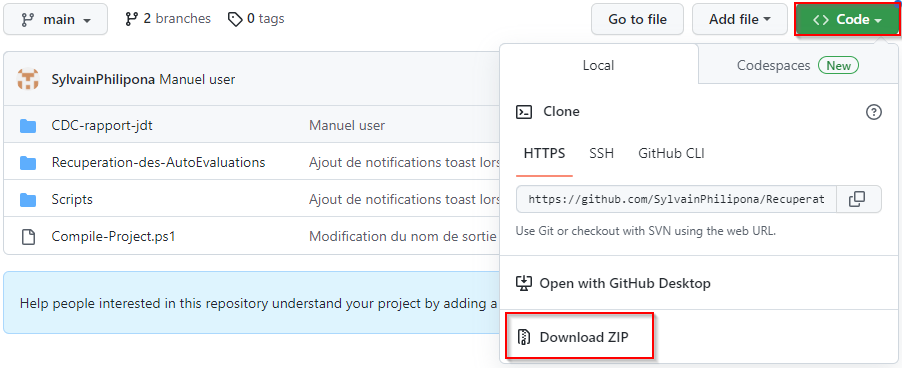


Figure Téléchargement du projet

Après avoir télécharger le projet en .zip, veuillez extraire le contenu de l’archive et vous rendre dans le dossier ‘’Recuperation-des-AutoEvaluations’’,

Puis faites clic-droit sur le fichier ‘’start.bat’’ puis ‘’Propriétés’’

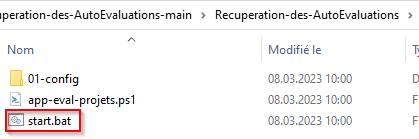


Figure Fichier de démarrage

Si vous avez l’option ‘’Débloquer’’ cliquez dessus. Sinon faites uniquement ‘’OK’’

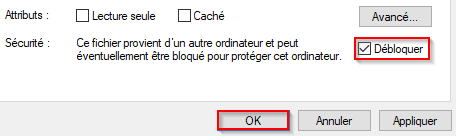


Figure Déblocage du fichier de démarrage

⚠ La première exécution du programme peux être particulièrement lente.

### Configurations

Dans un répertoire, mettez le dossier de scripts que vous avez téléchargé, puis renommez-le en lui attribuant le nom que vous désirez. Par exemple le nom du projet dont vous faites les auto-évaluations

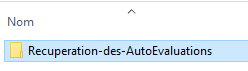
 🡪 

Figure Dossier du projet

Figure Dossier renommé

Ensuite entrez dans le dossier, vous trouverez le contenu suivant :

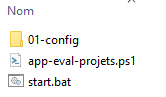


Figure Contenu du dossier

Rendez-vous dans le dossier ‘’01-config’’ puis ouvrez le fichier ‘’01-infos-proj-eleves.xlsx’’

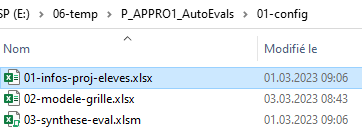


Figure Fichier de configuration

Dans l’onglet Excel ‘’students’’ entrez la liste des élèves dont vous voulez générer les auto-évaluations

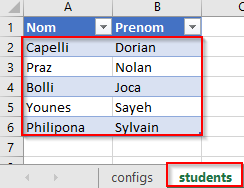


Figure Liste des élèves

Ensuite dans l’onglet ‘’configs’’ entrez les informations concernant le projet, puis enregistrez le fichier.

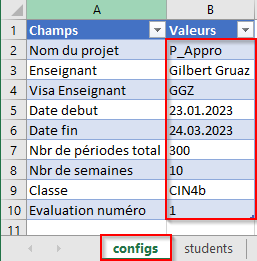


Figure Configurations du projet

### Création des auto-évaluations

Rendez-vous dans le dossier principal, puis double-cliquez sur le fichier ‘’start.bat’’. Ce fichier démarre le programme.

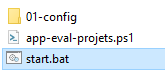


Figure Démarrage du programme

Une fois le programme démarré, cliquez sur ‘’Créer les auto-évaluations’’

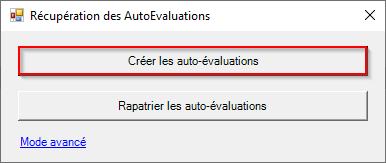


Figure Menu de création

Cette Pop-Up va apparaitre pour informer que cela peut prendre un peu de temps selon les performances de votre PC

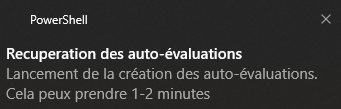


Figure Notification Toast

Une fois les auto-évaluations générées, ce message apparaitra. Cliquez sur ‘’OK’’

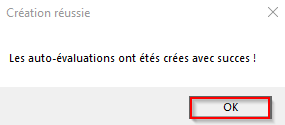


Figure Pop-up de résultat

Un dossier ‘’02-evaluations’’ sera créé contenant toutes les auto-évaluations.

Envoyez les auto-évaluations aux élèves pour qu’ils les remplissent.

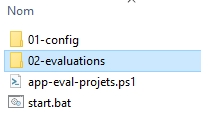
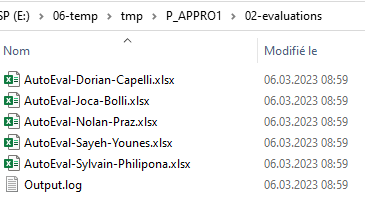
🡪

Figure Dossier de sortie

Figure Contenu du dossier de sortie

### Rapatriement des auto-évaluations

Une fois les auto-évaluations remplies et rendues par les élèves, remettez-les dans le dossier

‘’02-evaluations’’ puis relancez le script avec le fichier ‘’start.bat’’

Cette fois cliquez sur ‘’Rapatrier les auto-évaluations’’

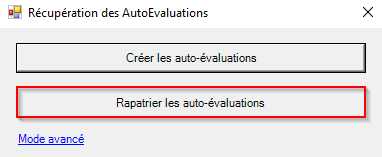


Figure Menu de rapatriement

Cette Pop-Up va apparaitre pour informer que cela peut prendre un peu de temps selon les performances de votre PC

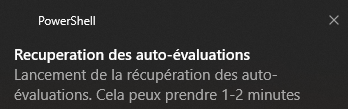


Figure Notification Toast

Une fois les auto-évaluations récupérées, ce message apparaitra. Cliquez sur ‘’OK’’

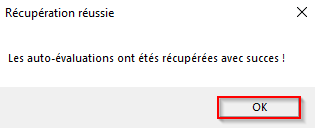


Figure Pop-up de résultat

Rendez-vous dans le dossier ‘’02-evaluations’’

Un fichier ‘’.xlsm’’ avec la synthèse des auto-évaluations sera créé. Ouvrez-le.

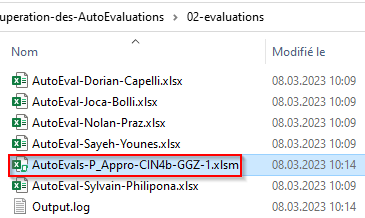


Figure Fichier de synthèse

Une fois ouvert pensez à activer les macros



Figure Activation des macros

Le fichier de synthèse contient un onglet pour chaque auto-évaluation, ainsi qu’un onglet ‘’MASTER’’

Faites les modifications désirées dans les auto-évaluations des élèves, puis une fois fini rendez-vous dans l’onglet ‘’MASTER’’



Figure Onglets lors de la synthèse

Dans l’onglet ‘’MASTER’’, cliquez sur ‘’Rapatrier les notes’’ et ‘’Export en PDF’’

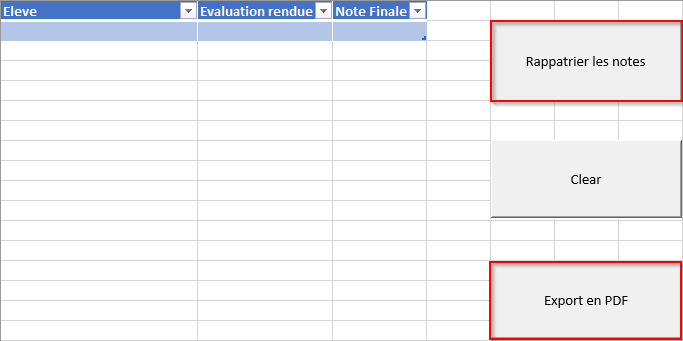


Figure Rapatriement et export des auto-évaluations

Les notes seront synthétisées dans le tableau et les auto-évaluations exportées en PDF dans le dossier ‘’02-evaluations’’

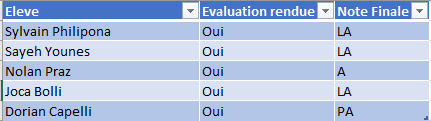


Figure Synthèse des notes

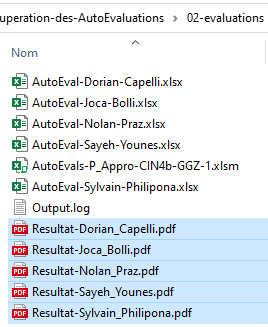


Figure Exports en PDF

## Archives du projet

Lien vers le répertoire Git du projet :

<https://github.com/SylvainPhilipona/Recuperation-des-AutoEvaluations>