



Table des matières

1	Analyse préliminaire	4
1.1	Introduction	4
1.2	Objectifs.....	4
1.3	Gestion de projet	4
1.4	Planification initiale	5
2	Analyse / Conception.....	6
2.1	Contexte produit	6
2.2	Contextes techniques	7
2.2.1	Opérationnel	7
2.2.2	Validation	7
2.2.3	Développement.....	8
2.3	Concept	9
2.4	Analyse fonctionnelle.....	10
2.4.1	Donner son Token à l'app.....	10
2.4.2	Changer de Repository et d'utilisateur	12
2.4.3	Afficher mes commits sous forme de JDT	14
2.4.4	Reprendre mon JDT sur un autre poste	15
2.4.5	Gérer les entrées	16
2.4.6	Exporter en PDF	22
2.5	Stratégie de test.....	23
2.6	Risques techniques	23
2.7	Planification	Erreur ! Signet non défini.
3	Réalisation.....	24
3.1	Points de design spécifiques	26
3.1.1	27
3.1.2	27
3.1.3	27
3.2	Déroulement	27
3.2.1	Sprints	27
3.2.2	Stories	29
3.3	Description des tests effectués.....	31
3.4	Bilan.....	31
3.4.1	Erreurs restantes	31
3.4.2	Stories	31
3.4.3	Dette technique.....	31
3.5	Recours à l'intelligence artificielle	33
3.6	Liste des documents fournis	34
4	Conclusions	34
5	Annexes.....	35
5.1	Résumé du rapport du TPI / version succincte de la documentation	35
5.2	Sources – Bibliographie.....	35
5.3	Journal de travail	35
5.4	Manuel d'Installation	35

5.5	Manuel d'Utilisation.....	35
5.6	Archives du projet.....	35

NOTE L'INTENTION DES UTILISATEURS DE CE CANEVAS :

Toutes les parties en italiques sont là pour aider à comprendre ce qu'il faut mettre dans cette partie du document. Elles n'ont donc aucune raison d'être dans le document final.

De plus, en fonction du type de projet, il est tout à fait possible que certains chapitres ou paragraphes n'aient aucun sens. Dans ce cas il est recommandé de les retirer du document pour éviter de l'alourdir inutilement.

1 Analyse préliminaire

1.1 Introduction

Le but de ce projet est de créer une application qui permet aux utilisateurs de générer un journal de travail en utilisant les commit GitHub. Son nom est GitJournal. L'application se base sur la date des commits, la personne ayant effectué le commit, les métadonnées du commit. Les métadonnées comprennent le titre du commit et la description du commit.

GitJournal va donc permettre de gagner beaucoup de temps car il permet de ne pas perdre de temps à remplir son journal de travail. De plus, il permet d'exporter le JDT en pdf avec une belle mise en page et bien présentable.

Ce projet est très similaire à [ETML-INF/gitjournal](https://github.com/ETML-INF/gitjournal) qui est fait en Rust. Ce projet lui est une application desktop développée en WPF et .Net 8.

1.2 Objectifs

Objectifs de formation :

Ce projet et mon projet de TPI.

Objectifs du produit :

Ce produit doit permettre d'exporter un JDT en format PDF en se basant sur les commits dans une repository donnée.

Il va également permettre à l'utilisateur de modifier les données des commits en supprimer et même en ajouter pour refléter au mieux le travail réalisé.

On peut exporter nos modifications dans un fichier et le réimporter sur un autre poste pour pouvoir transmettre notre rapport au format gitj ou travailler sur plusieurs postes.

1.3 Gestion de projet

Pour ce projet, je vais utiliser la méthode agile, elle permet une gestion plus flexible du temps et des tâches et permet grâce au sprint review et user stories validée par le CP de ne pas s'égarer.

Il y aura au total 2 sprints durant ce projet, ils vont durer approximativement 2 semaines.

Lors de ses sprints, il y aura une sprint review avec le chef de projet pour faire un point sur la progression du projet.

Lors du début de chaque journée/ demi-journée, je fais un résumé de ce que j'ai fait la fois d'avant et ce que je compte faire aujourd'hui.

Outils :

Pour la planification et le management des user stories, j'utilise GitProject et les issues GitHub.

Pour le versioning, j'utilise GitHub.

Les maquette et schéma sont fait sur Figma et DrawIO.

J'ai utilisé la suite office pour le rapport et mon JDT.

Le JDT est fait sur un canevas de JDT crée par l'ETML ce qui simplifie la chose pour moi.

1.4 Planification initiale

No	Objectifs	Durée	Dates	Date Sprint Review	
1	Donner son Token à l'app	~2H	~46H30	05.05 Au 16.05	Vendredi 16.05.2025 9H00
	Changer de Repository	~10H30			
	Afficher mes commits sous forme de JDT	~12H			
	Reprendre mon JDT sur un autre poste	~10H			
	Documentation	~10H			
2	Gérer les entrées	~10H50	~36H50	05.05 Au 16.05	Vendredi 23.05.2025 9H00
	Exporter en PDF	~18H			
	Documentation	~8H			

	28.04 - 02.05 Après Matin Midi	05.05 - 09.05 Après Matin Midi	12.05 - 16.05 Après Matin Midi	19.05 - 23.05 Après Matin Midi	26.05 - 30.05 Après Matin Midi	02.06 - 06.06 Après Matin Midi
Lundi	-	05:35	05:35	05:35	03:10 Examens	03:10 -
Mardi	-	-	-	-	-	-
Mercredi	-	07:15	07:15	07:15	07:15	-
Jeudi	-	- 03:10	- 03:10	- 03:10	Férieré	-
Vendredi	07:15	07:15	07:15	07:15	Férieré	-

Le total d'heures dans ma planification n'atteint pas les 90H prévue pour ce projet. Dans ces heures, je n'ai pas pris en compte le vendredi 2 mai 2025. C'est le jour où j'ai reçu mon CDC et j'ai pris la décision de ne pas l'inclure dans mon premier sprint. Ce vendredi a duré 7H15.

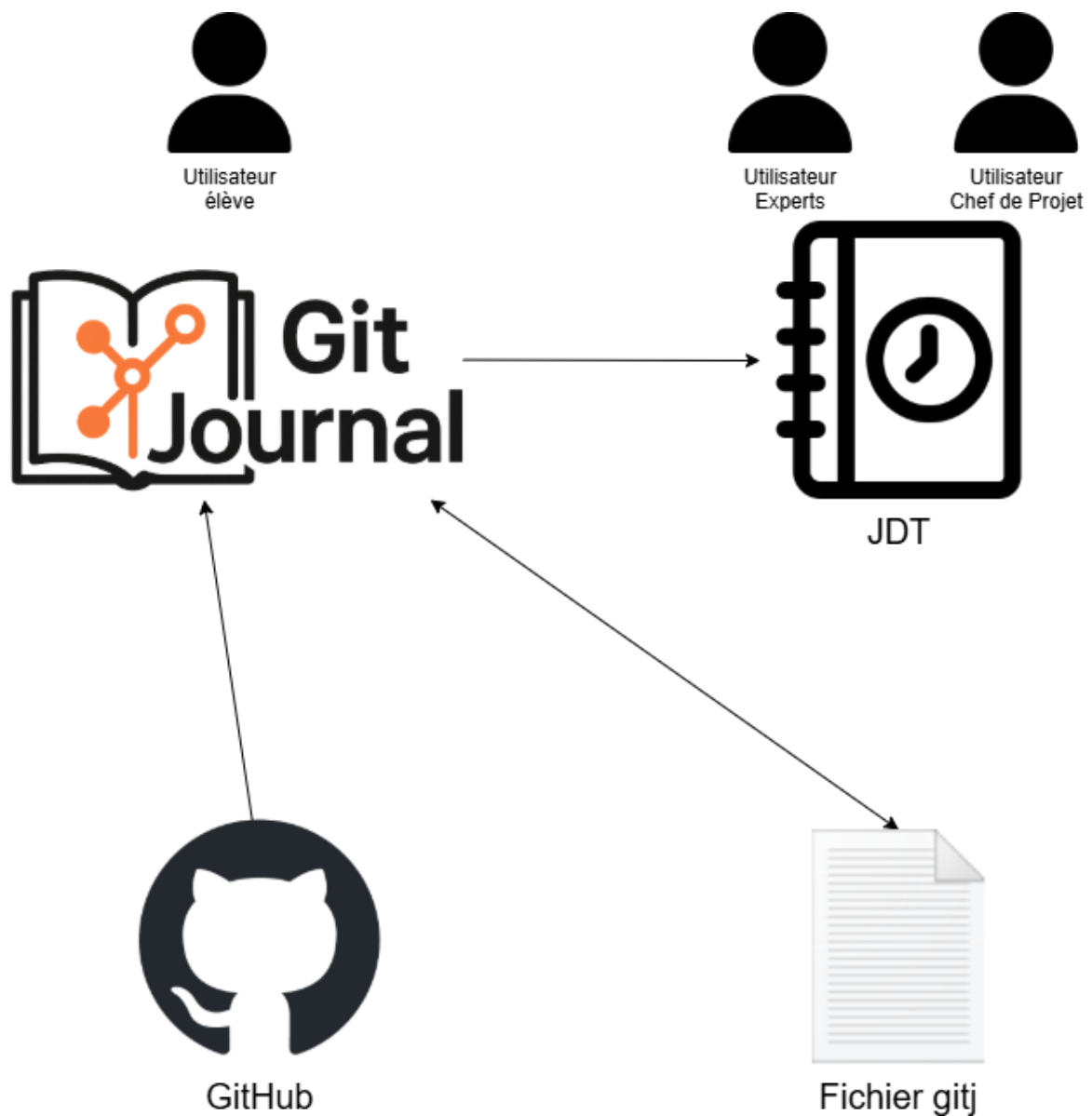
2 Analyse / Conception

2.1 Contexte produit

GitJournal est un program desktop fait pour Windows (10 et 11).

Il est fait pour être utilisé par un seul utilisateur à la fois. Comme l'application requiert un PAT pour se connecter à github, un seul token peut être introduit à la fois ce qui rend l'application mono utilisateur.

Le système exploite les métadonnées des commits GitHub et prend en compte les modifications de l'utilisateur enregistrées dans un fichier gitj.



2.2 Contextes techniques

2.2.1 Opérationnel

Pour le contexte opérationnel, GitJournal est exécuté sur n'importe quelle machine sous Windows 10 ou 11.

Avec ou sans droits administrateurs.

Accès à internet et spécialement GitHub, peu importe le lieu, si on veut récupérer des informations depuis GitHub, on va avoir besoin de connexion et que GitHub soit accessible pour afficher des informations de repository ou pour actualiser un fichier gitj. Pour ouvrir un fichier gitj, il faut juste l'application GitJournal.

GitJournal fait appel à l'API de GitHub via PAT (Personal access token).

L'application utilise un installer (exe ou msi) pour installer l'application.

Si on essaye d'ouvrir un fichier .gitj, il s'ouvre automatiquement avec GitJournal

2.2.2 Validation

Les tests de validation se passent sur des PC sous Windows 10 (22H2), excepté ils ne sont pas le poste où se passe le développement.

Un des postes sera

Les tests sont effectués sur au moins 2 postes.

Il n'y aura pas de droits administrateurs.

Accès à internet et spécialement GitHub.

GitJournal fait appel à l'API de GitHub via PAT (Personal access token)

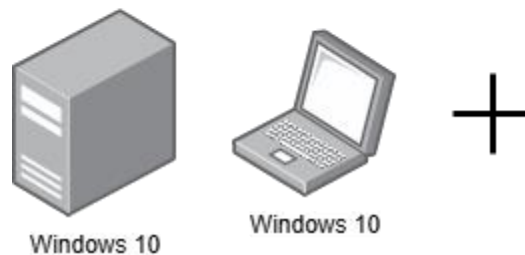
Pour ces tests il faut un repository de quelqu'un qui a comme moi formater son repository de façon que GitJournal soit capable d'utiliser les metadata pour créer un JDT complet. Cela importe peu à qui appartient le repository, il peut-être le repo de GitJournal. Il faut juste que le PAT soit celui du propriétaire du repository.

Le code est cloné depuis de repository GitHub.

Aucun fichier en plus du code et du fichier de config de l'application (si il y en a un) ne sont présents.

L'application est lancée depuis un exécutable (sans installeur) généré au préalable sur la machine de dev.

Le programme sera fait en .Net 8, il faudra donc que la machine ait bien .Net 8 d'installé.



2.2.3 Développement

Le développement se passe sur un PC sous Windows 10 (22H2).

Il n'y aura pas de droits administrateurs.

Accès à internet et spécialement GitHub.

GitJournal fait appel à l'API de GitHub via PAT (Personal access token)

Pour les données, je vais utiliser le repository pour ce projet car j'ai fait en sorte de formater mes commits pour qu'ils soient utilisable par GitTools.

Liens vers le repository : <https://github.com/Morgan-DD/GitJournal>

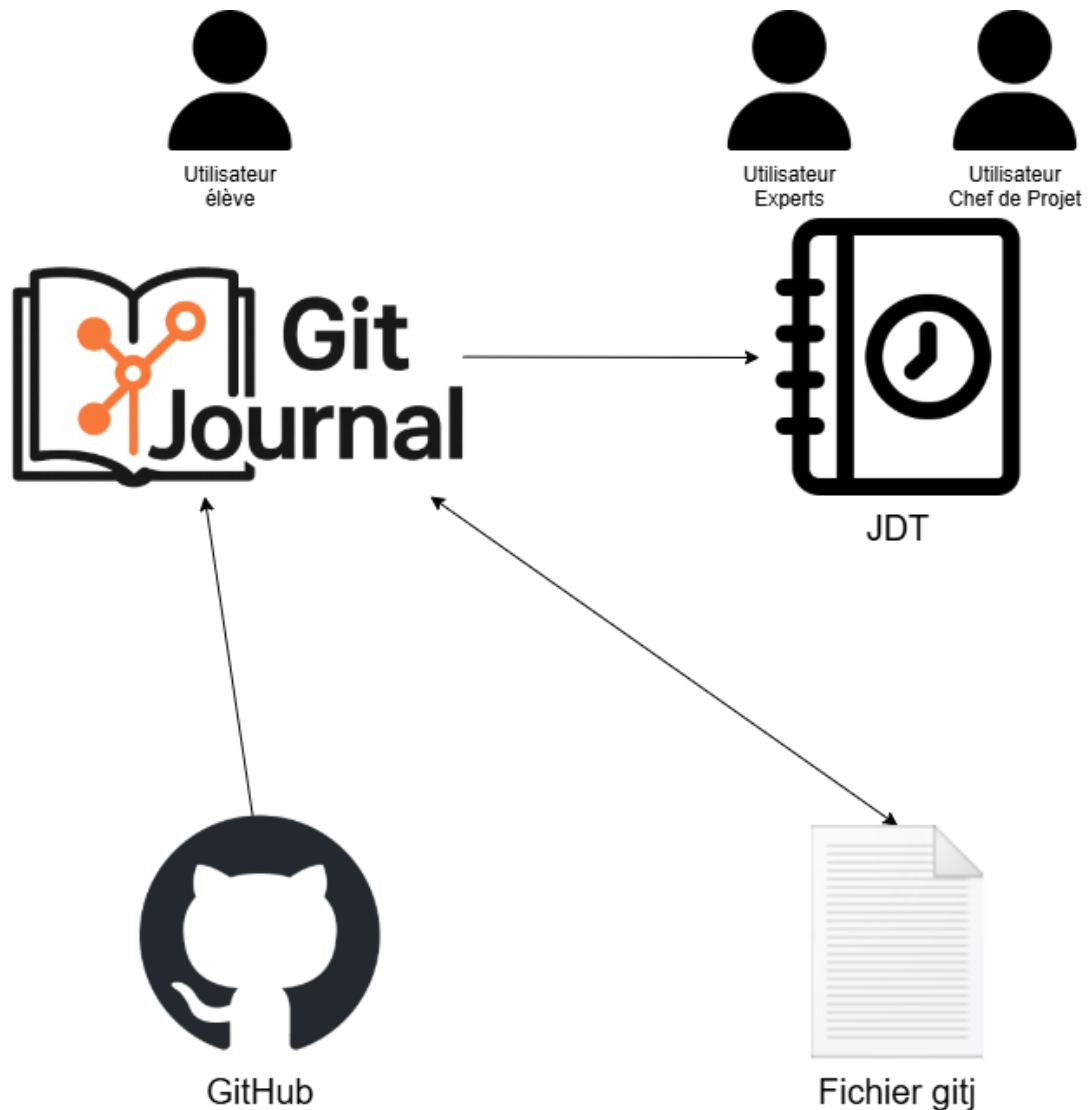
Pour passer de l'environnement de développement à l'environnement de validation, il n'y a pas grand-chose à faire excepté que le repository soit à jour.

Le développement se passera sur Visual Studio 2022.

Le programme sera fait en .Net 8, il faudra donc que la machine ait bien .Net 8 d'installé.

J'ai choisi .Net 8 car dans les dernières versions de .Net c'est la seule maintenue à long terme (.Net 9 Standard Support Time) et que j'ai déjà utilisé .Net 8 pour mon préTPI donc je suis familier avec cette version de .Net.

2.3 Concept



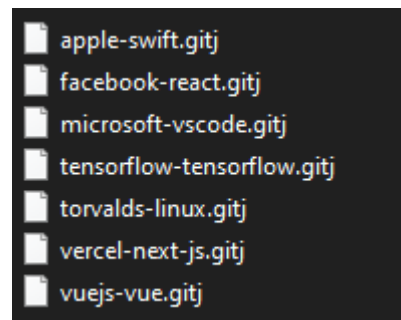
Les données des commits peuvent être récupérées depuis GitHub ou le fichier gitj.

Il y a un fichier gitj par repository, ils sont tous stockés dans un dossier repos par exemple.

Il est possible de partager un fichier gitj et de l'importer comme ça les utilisateurs n'ayant pas accès au repository (si le repo est privé) peuvent aussi lire le JDT directement depuis GitJournal et même le modifier.

En revanche, il ne sera pas mis à jour car étant incapable de synchroniser les informations avec celles de GitHub.

Il est aussi possible d'exporter le JDT en PDF pour une lecture du JDT sans GitJournal installé.

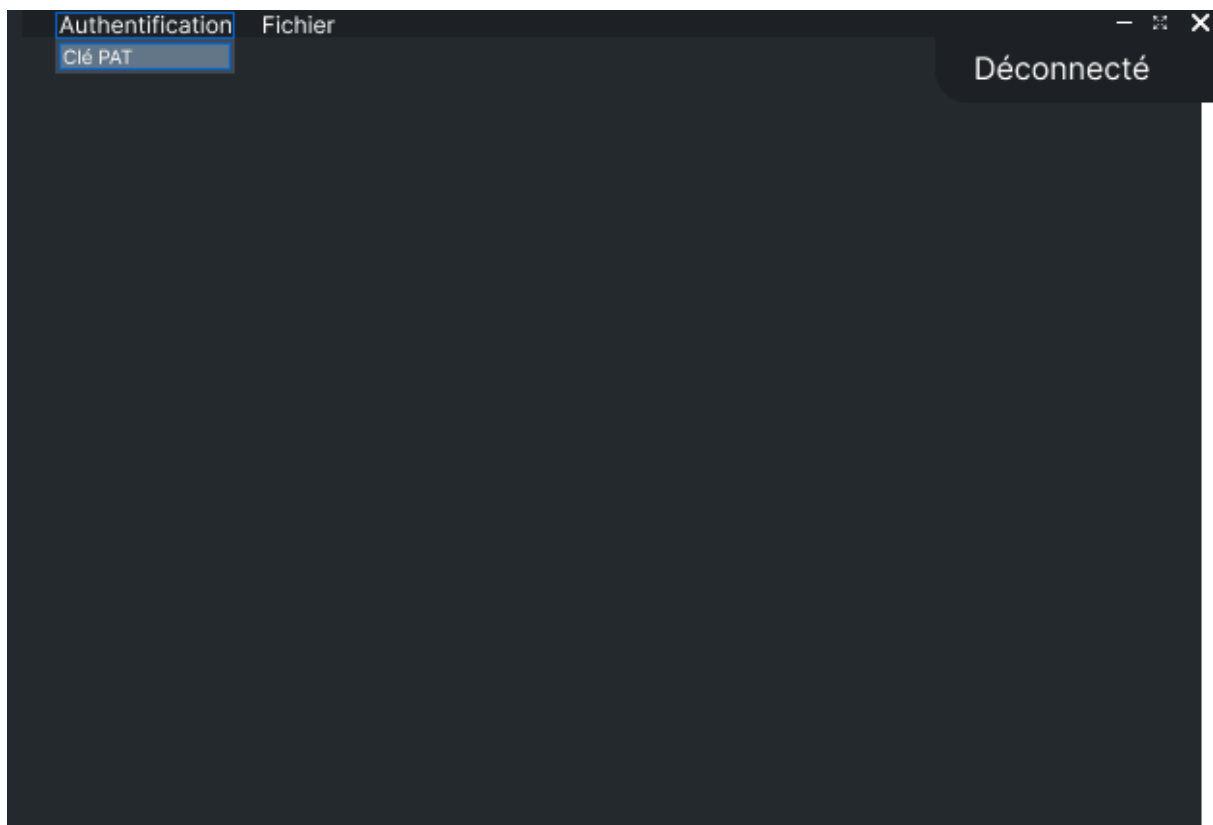


2.4 Analyse fonctionnelle

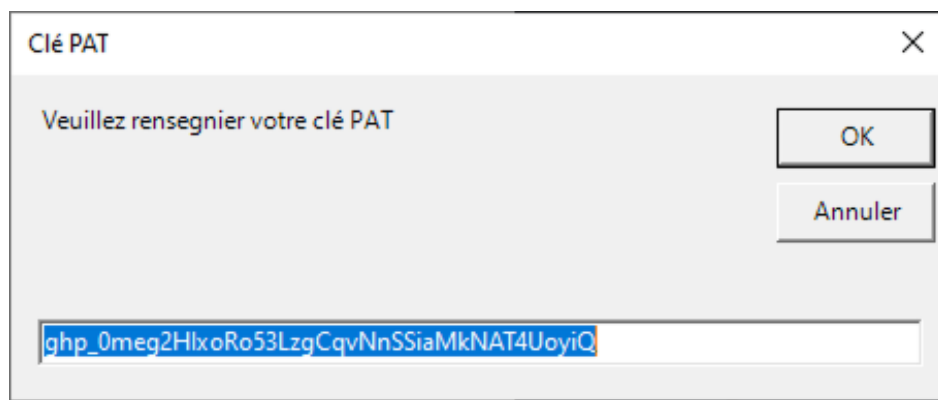
2.4.1 Donner son Token à l'app

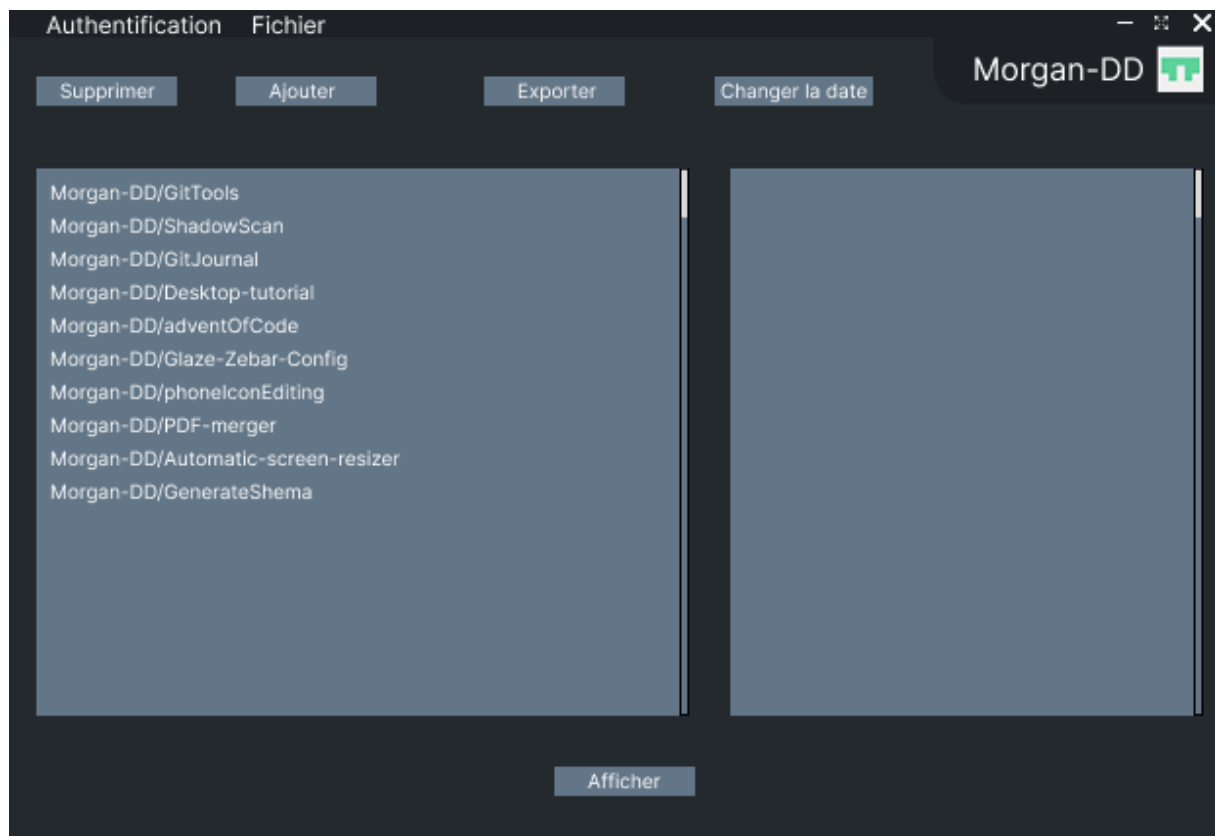
En tant qu'utilisateur, je veux pouvoir m'authentifier en passant par le menu de l'application.

Je clique en haut sur l'option Authentification, puis sur Clé PAT.



Une fenêtre va s'ouvrir et dans celle si, je vais renseigner ma clé PAT.





Test :

Je lance GitJournal, en haut à droite je vois déconnecté

Dans le menu, j'appuie sur Authentification -> Clé PAT. Une fenêtre apparaît

J'introduis ma clé GitHub, un popup apparaît pour notifier de la réussite de l'action

J'introduis une clé bidon, un popup apparaît pour notifier de l'échec de l'action

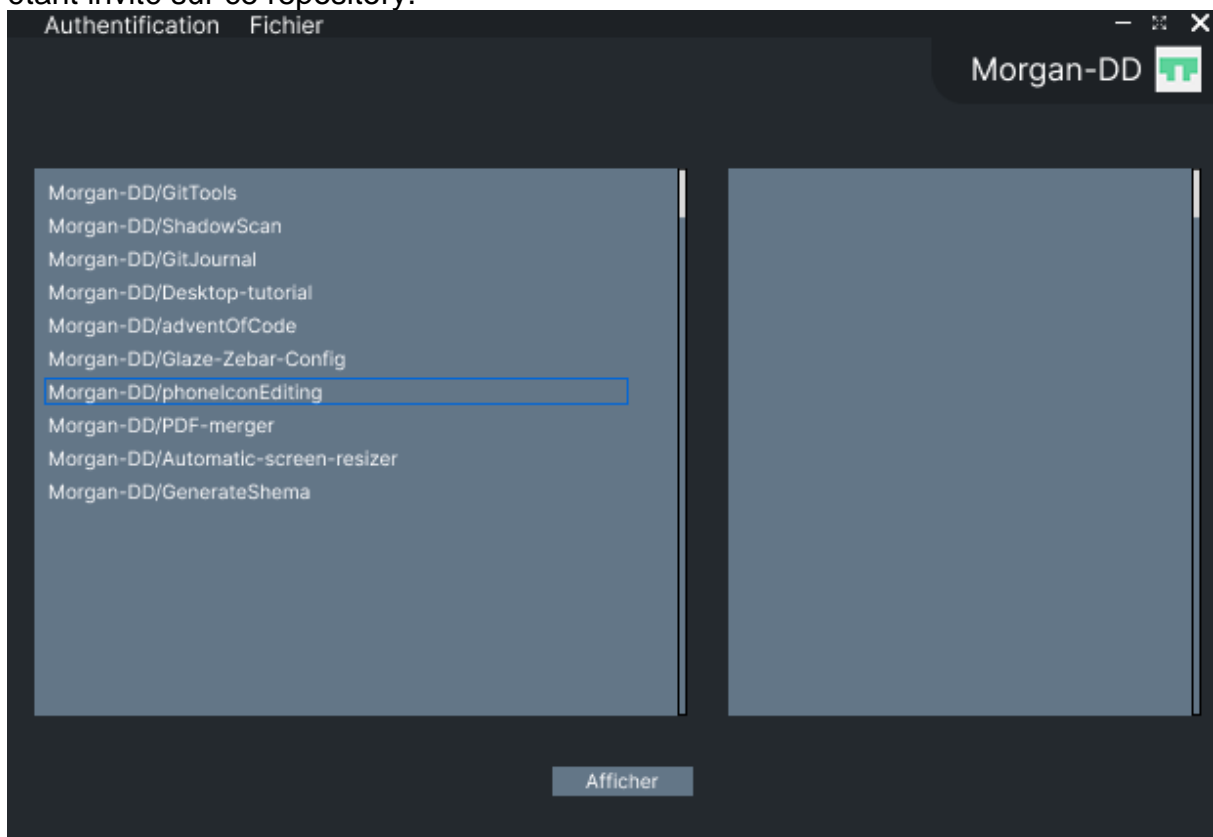
Dans le menu en dessous, un tableau avec la liste de mes repositories apparaît

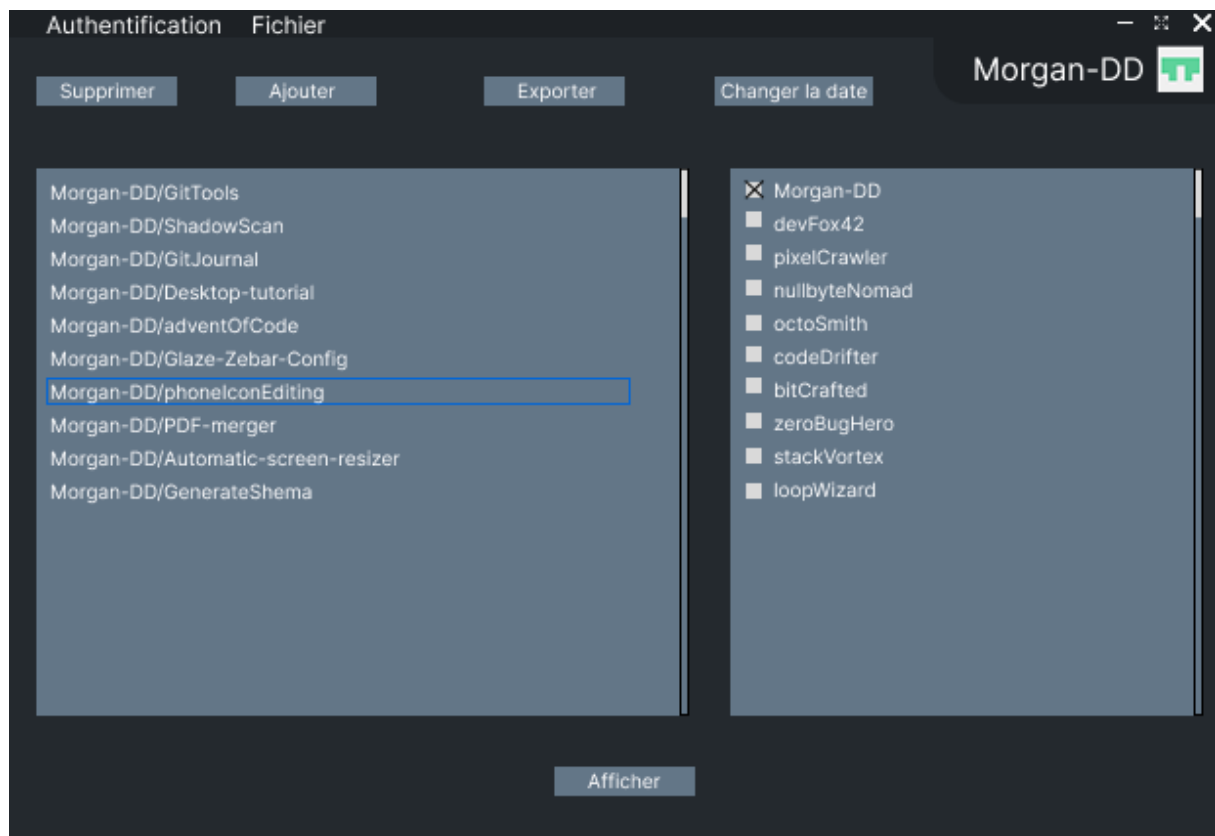
2.4.2 Changer de Repository et d'utilisateur

Une fois connecté, en tant qu'utilisateur, je veux pouvoir choisir et changer avec lequel repository Gitjournal va générer un JDT.

Dans le menu de gauche, je vois tous mes repository et ceux dans lesquels je suis invité.

Une fois le repository choisi, dans le menu de droite apparaissent les utilisateurs étant invité sur ce repository.





Test :

Je sélectionne un repository dans la liste des repositories. Dans la liste des utilisateurs, toutes les personnes invitées dans le repo apparaissent ainsi que le propriétaire du repo








Je coche un ou plusieurs utilisateurs dans la liste

2.4.3 Afficher mes commits sous forme de JDT

En tant qu'utilisateur, je veux pouvoir afficher tous les commit de mon repository sous forme de JDT.

Je veux avoir ces informations pour chaque entrée dans le JDT :

Titre, détails sur l'action effectuée, le nom de l'utilisateur qui l'a réalisé, savoir si c'est toujours en cours ou si c'est une tâche finie et la durée de la tâche.

Authentification Fichier		Morgan-DD 		
Supprimer	Ajouter	Exporter	Changer la date	Changer de repo
Titre	Contenu	Utilisateur	Status	Durée
20 Janvier 2027				
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
■ Doc - JDT	Mise à jour du JDT	Jhon Doe	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
■ Doc - JDT	Mise à jour du JDT	Chuck	Done	10 min
Total				1H
19 Janvier 2027				
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
Grand Total				89H

Test :

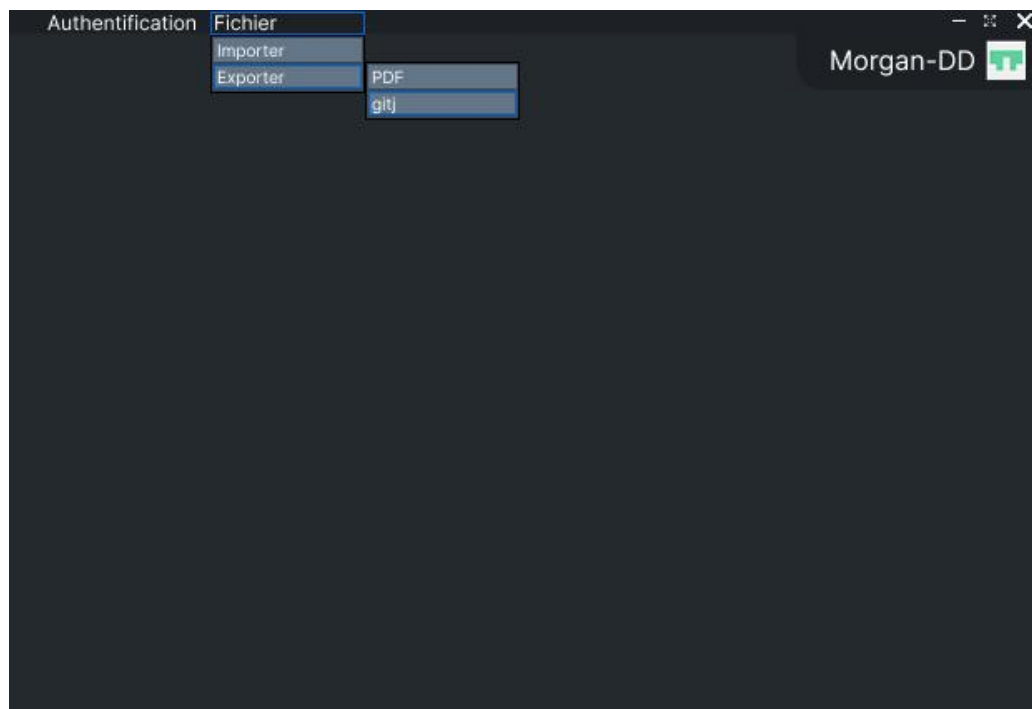
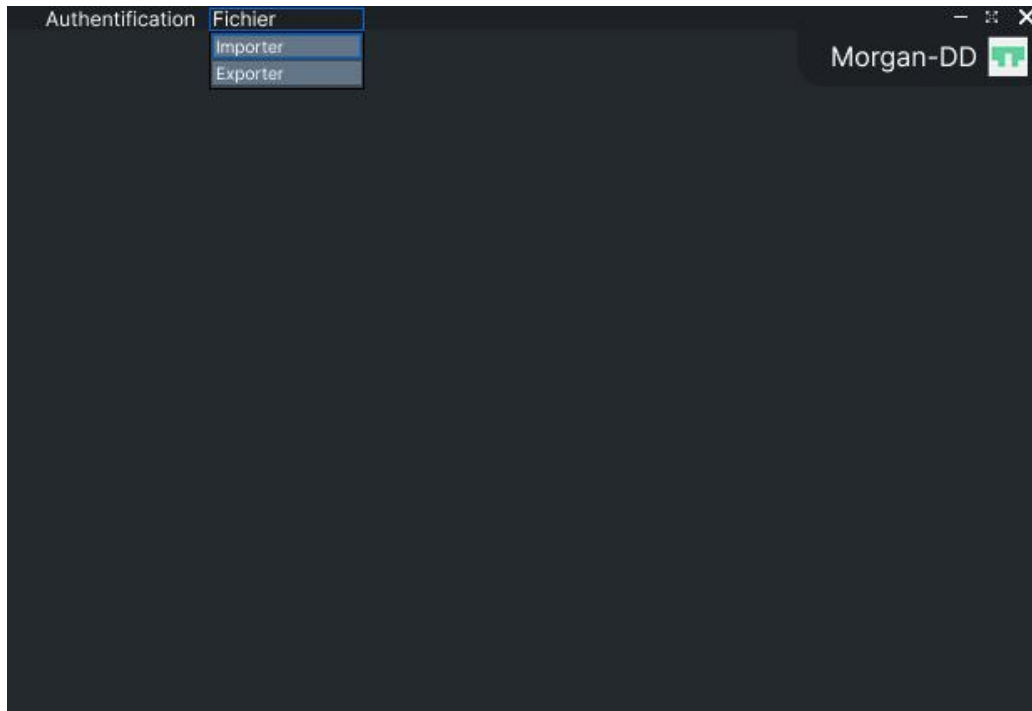
Je sélectionne un repository ainsi que deux utilisateurs puis j'appuie sur Afficher. Il y a plusieurs tableaux qui apparaissent en dessous, un par jour. Je peux défiler dans cette liste de tableaux.

Je choisis un seul utilisateur et la colonne User n'apparaît pas

2.4.4 Reprendre mon JDT sur un autre poste

En tant qu'utilisateur, je veux pouvoir faire des modifications sur une storie sur un de mes repository. Je change de salle et je peux reprendre mes modifications.

Je veux être capable d'exporter et importer un fichier qui me permet de sauvegarder mes modifications pour un repository donné. Je avoir cette option disponible en haut dans le même menu que celui ou je donne la clé PAT.



Test :

J'effectue des modifications sur un repository. Je clique sur Fichier, Export, je sauvegarde mon fichier sur une clé USB. Sur cette clé, je retrouve mon fichier gitj.


Je me rends sur un autre poste avec ce fichier que j'ai exporté. Je lance GitTools, en haut dans le menu je presse sur Fichier puis Importer, je choisis mon fichier gitj puis je retrouve mes modifications.

2.4.5 Gérer les entrées

En tant qu'utilisateur, je veux pouvoir modifier, ajouter et supprimer certaines informations dans le JDT.

Le titre, contenu, statut, temps et la date de l'entrée doivent être modifiables.

En cliquant dans un champ, ça me permet de le modifier et sauvegarde automatiquement.

Authentification Fichier		Morgan-DD 		
Supprimer Ajouter Exporter Changer la date Changer de repo				
Titre	Contenu	Utilisateur	Status	Durée
20 Janvier 2027				
Doc - JDT		Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Jhon Doe	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Chuck	Done	10 min
Total				1H
19 Janvier 2027				
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Grand Total				89H

Authentification

Fichier

Supprimer

Ajouter

Exporter

Changer la date

Changer de repo

Morgan-DD

Titre	Contenu	Utilisateur	Status	Durée
20 Janvier 2027				
Doc - JDT	Remise en forme du JDT (erreurs d'horraire)	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Jhon Doe	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Chuck	Done	10 min
Total				1H
19 Janvier 2027				
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à iour du JDT	Morqan -DD	Done	10 min
Grand Total				89H

En cochant la checkbox à gauche des entrées, on peut ensuite cliquer sur [Changer la date] pour pouvoir changer la date de l'entrée, ce qui va la changer de tableau.

Authentification

Fichier

Supprimer


Ajouter

Exporter

Changer la date

Changer de repo

Morgan-DD



Titre	Contenu	Utilisateur	Status	Durée
20 Janvier 2027				
Doc - JDT	Remise en forme du JDT (erreurs d'horraire)	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Jhon Doe	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Chuck	Done	10 min
Total				1H
19 Janvier 2027				
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à iour du JDT	Morqan -DD	Done	10 min
Grand Total				89H

Date
05.10.2027

Annuler
Ajouter

Authentification
Fichier
Morgan-DD

Supprimer
Ajouter
Exporter
Changer la date
Changer de repo

Titre	Contenu	Utilisateur	Status	Durée
20 Janvier 2027				
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Jhon Doe	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Total				1H
19 Janvier 2027				
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Grand Total				89H

Toujours en cochant la checkbox, en appuyant sur le bouton supprimer, cela supprime l'entrée.

Authentification

Fichier

En appuyant sur le bouton [Ajouter] en haut, une fenêtre apparait nous demandant des informations pour créer une entrée, (toutes sont obligatoires sauf le contenu). Une fois renseignées, cela crée l'entrée à la bonne date.

Authentification Fichier

Morgan-DD

Supprimer


Ajouter




Exporter

Changer la date

Changer de repo

Titre	Contenu	Utilisateur	Status	Durée
20 Janvier 2027				
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Jhon Doe	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Total				1H
19 Janvier 2027				
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
■ Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Grand Total				89H

Titre	Meeting
Text	Meting avec chuck noris
Personne	Morgan-DD 
Status	<input checked="" type="radio"/> DONE <input type="radio"/> WIP
Durée	15 Min
Date	05.6.2027
<div>Annuler</div> <div>Ajouter</div>	

Authentification Fichier		Morgan-DD 		
<input type="button" value="Supprimer"/> <input type="button" value="Ajouter"/> <input type="button" value="Exporter"/> <input type="button" value="Changer la date"/> <input type="button" value="Changer de repo"/>				
Titre	Contenu	Utilisateur	Status	Durée
20 Janvier 2027				
<input type="checkbox"/> Doc - JDT	Remise en forme du JDT (erreurs d'horraire)	Morgan -DD	Done	10 min 
<input type="checkbox"/> Doc - JDT	Mise à jour du JDT	Jhon Doe	Done	10 min
<input type="checkbox"/> Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
<input type="checkbox"/> Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
<input type="checkbox"/> Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
<input type="checkbox"/> Doc - JDT	Mise à jour du JDT	Chuck	Done	10 min
<input type="checkbox"/> Code	Ajout de la fonction pour la sécurité du login	Chuck	Done	3H
Total				4H
19 Janvier 2027				
<input type="checkbox"/> Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
<input type="checkbox"/> Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
<input type="checkbox"/> Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min 
Grand Total				89H

Test :

Je choisi une entrée qui viens depuis GitHub (en vert), je modifie la valeur de cette entrée puis sauvegarde. Le champ devient jaune (signifie la modification).

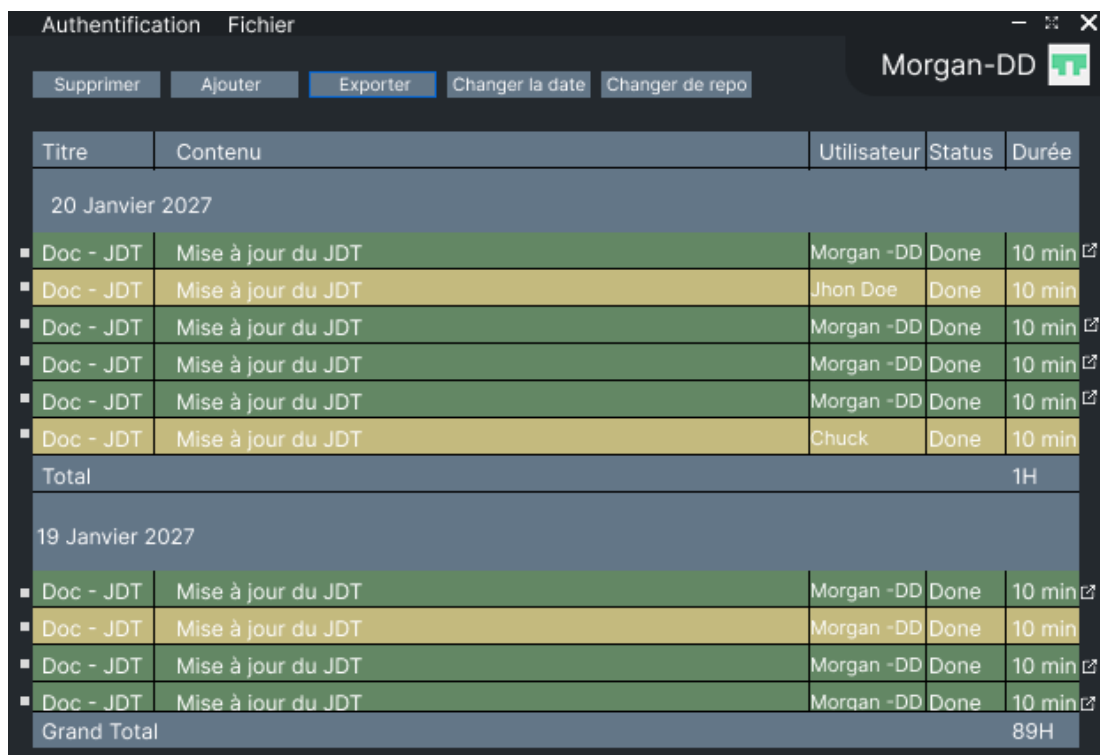
Je choisi une ou plusieurs entrées, je coche la checkbox à droite. Je presse le bouton Supprimer. L'entrée disparaît.

Je choisi une ou plusieurs entrées, je coche la checkbox à droite. Je presse le bouton Modifier. Je choisis une autre date. L'entrée va maintenant apparaître dans le tableau du jour choisi.

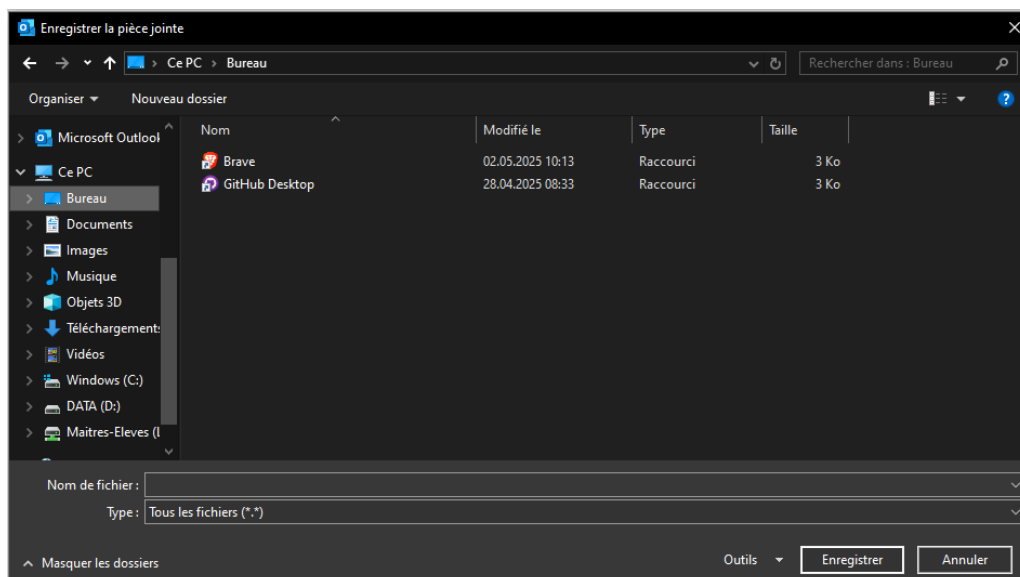
Je presse le bouton Ajouter. Une fenêtre va apparaître avec plusieurs champs à remplir. Je remplis les champs obligatoires puis presse sur ajouter. Dans le tableau du jour que j'ai choisi apparaît ma nouvelle entrée créée à la main. Elle est jaune

2.4.6 Exporter en PDF

En tant qu'utilisateur, je veux pouvoir exporter mon JDT en PDF. Dans le menu en haut sous fichier, Exporter, je choisis Exporter en PDF. Je choisis le chemin et GitJournal me génère un fichier PDF qui est mon JDT.



Titre	Contenu	Utilisateur	Status	Durée
20 Janvier 2027				
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Jhon Doe	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Chuck	Done	10 min
Total				1H
19 Janvier 2027				
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Doc - JDT	Mise à jour du JDT	Morgan -DD	Done	10 min
Grand Total				89H



Test :

Dans le menu en haut de page, je presse fichier -> Exporter -> Exporter en PDF, choisis le dossier où exporter le fichier PDF. Dans ce dossier, un PDF est généré.

2.5 Stratégie de test

Les tests seront faits sur plusieurs postes différents pour garantir une certaine sécurité quant au fonctionnement des features testées.

Les tests seront simplement les tests de validation des différentes user stories listée dans le point [Analyse fonctionnelle](#).

Les tests seront faits sans donnée mise en place au préalable sauf si un test d'une des user stories dis le contraire.

Le PAT fait exception à cette règle car sans lui, il est impossible de récupérer les informations depuis GitHub.

Le test peut être effectué par toute personne ayant un repository qui a été formaté pour fonctionner avec GitJournal, c'est-à-dire ajouter dans la description des commits la durée de chaque tâche entre crochets : [1h30], [5min], [3h], ect..
Il faut également que dans la description après l'information de la durée apparaisse le statut [DONE] ou [WIP] (Work In Progress) pour que le commit soit affiché dans GitJournal et dans le fichier PDF généré.

Pour plus de détails sur le contexte de validation, voir section [2.2.2 Validation](#).

2.6 Risques techniques

GitHub :

Comme GitJournal s'appuie sur l'API de GitHub, si l'api ou tout GitHub est down, ça Paralyserait le projet.

Malheureusement, je n'ai pas de moyen de me protéger de ce type de panne car tout mon projet tourne autour de GitHub.

PDF

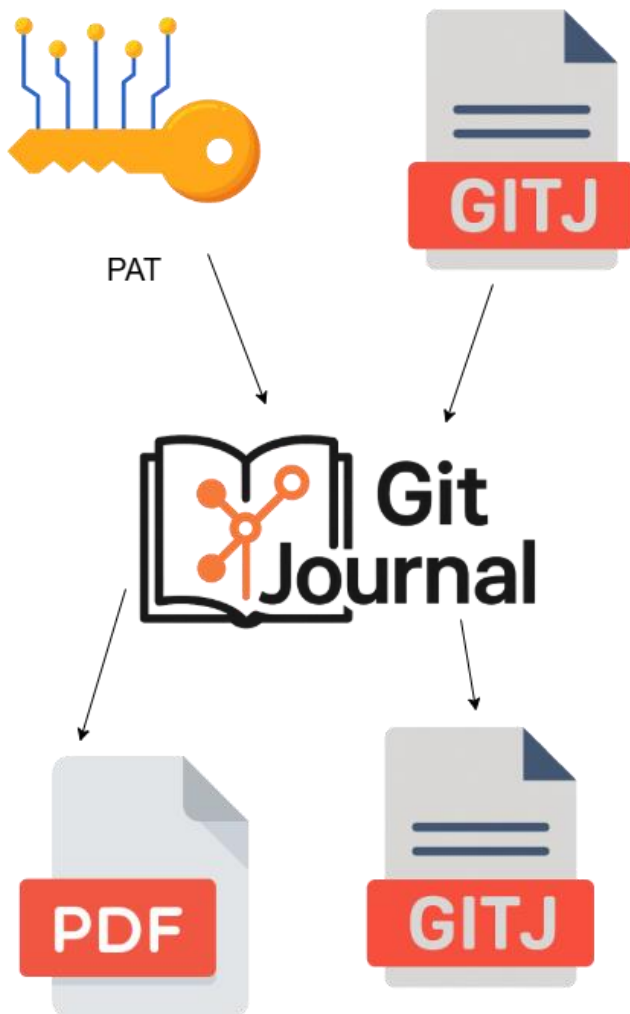
Je n'ai pas encore utilisé le c# pour manipuler des PDF (partie export du JDT en PDF), cela pourrait prend plus de temp que prévu ou même qui ça me bloque complètement.

Pour me protéger, j'ai alloué un temp important pour cette partie du projet qui est pour moi la partie la plus complexe.

3 Réalisation

3.1 Gestion des différents fonctionnements

GitJournal a deux manières de fonctionner, avec le PAT ou un fichier gitj.



Pour le PAT :

GitJournal va récupérer tous les repositories dont l'utilisateur est propriétaire et ceux sur lesquels il est invité également et tout cela une fois s'être connecté avec un PAT valide ayant au minimum les droits sur les repos.

Il peut ensuite choisir un repo ainsi qu'un utilisateur pour générer un JDT.

Si c'est la première fois que l'utilisateur génère un JDT à partir de ce Repo,
Le JDT généré est sauvegarder dans un fichier gitj créer automatiquement par le programme.

Au contraire si le fichier existe déjà, GitJournal appond les nouvelles informations (commits) au fichier.

Pour les modifications, il va écrire les modifications dans le fichier gitj qui à ce moment existe à coup sûr.

Pour le fichier gitj :

Quand un utilisateur sans être connecté ouvre un fichier gitj à l'aide de GitJournal, le fichier est simplement affiché, ce qui est plutôt rapide car il ne va pas chercher sur github de potentiels nouveautés.

Quand l'utilisateur effectue des modifications sur ce fichier, les changements seront effectués directement dans le fichier d'origine.

3.2 Enregistrement et gestion des changements

GitJournal reçoit des informations depuis 2 source distinctes, GitHub et du fichier gitj si il existe.

Les données sont stockées dans un tableau d'un objet custom nommé Commit_Info.

```
public List<Commit_Info> _commits { get; set; }

public class Commit_Info
{
    11 references
    public string CommitId { get; set; }
    7 references
    public string Title { get; set; }
    8 references
    public string Content { get; set; }
    8 references
    public string User { get; set; }
    6 references
    public string Status { get; set; }
    8 references
    public TimeSpan Duration { get; set; }
    3 references
    public bool ExistingStatus { get; set; }
    7 references
    public DateTime Date { get; set; }
    2 references
    public string Url { get; set; }
    3 references
    public string Origin { get; set; }

    3 references
    public bool IsTitleModified { get; set; }
    3 references
    public bool IsContentModified { get; set; }
    2 references
    public bool IsUserModified { get; set; }
    3 references
    public bool IsStatusModified { get; set; }
    3 references
    public bool IsTDurationModified { get; set; }
}
```

En premier GitJournal vérifie si un fichier gitj pour le repo actuel existe, si oui il récupère les données venant de ce fichier et les ajoute à ce tableau.

Après cette vérification, que le fichier existe ou pas, GitJournal va chercher les commit dans GitHub et les ajoute au tableau.

Si il y a un commit à double (CommitId fonctionne comme clé primaire) alors GitJournal priorise les commits venant du fichier gitj. Les deux types de commits sont différenciable à l'aide de la valeur de Origin "GitJ" ou "GitHub". Donc les commits dont l'origine est "GitJ" sont préférés à ceux dont l'origine est "GitHub".

Si les deux commits en conflits ont la même origine, GitJournal priorise le plus récent des deux.

Une fois cette étape passée, GitJournal va enregistrer dans le fichier gitj le tableau _commits sous forme de json. Il va changer l'origine de tout les commits qu'il enregistre en "GitJ".

3.3 Gestion du PAT

Une fois connecté, manuellement car première connexion ou connexion automatique car déjà connecté, GitJournal va récupérer les noms de repositories auquel le PAT lui donne accès.

Quand on c'est connecté une fois, le PAT est sauvegardé dans un fichier nommé GitJournal.Debug.

ATTENTION !! Le PAT est écrit en claire dans ce fichier ce qui n'est pas du tout une bonne pratique. J'ai fait ça de cette manière car trouver et mettre en place un moyen vraiment efficace pour sécuriser ce PAT est une tâche longue et complexe. Malheureusement je n'ai pas le temp de mettre en place cette bonne pratique car le temp me fait défaut. J'avais planifié dès le début que le stockage du PAT serait lacunaire.

3.4 Points de design spécifiques

Ce chapitre est constitué de plusieurs sous-chapitre.

Chaque sous-chapitre explique un point de design technique particulier, quelque chose que vous avez dû inventer pour répondre au besoin et qui ne peut pas s'expliquer par de simples commentaires dans le code.

Il s'agit d'explications techniques sur le fonctionnement du système. Les explications sont appuyées par des diagrammes, ou de très brefs éléments de code.

NE PAS mettre ici des pratiques usuelles que tout professionnel de la branche connaît déjà. Par exemple, n'EXPLIQUEZ PAS ICI CE QU'EST LE PATTERN MVC.

Exemple (simplifié à l'extrême) : Protection contre des formulaires mal intentionnés ou modifiés

- *Au moment de générer le formulaire, le script php :*
 - *Concatène les noms de tous les champs contenus dans le formulaire*
 - *Calcule un hash SHA256 de la chaîne obtenue*
 - *Ajoute un input nommé « CSRF » de type hidden dans le form*
- *A la réception du POST du formulaire*
 - *Concatène les noms des indices de \$_POST*
 - *Calcule un hash SHA256 de la chaîne obtenue*
 - *Vérifie que la valeur du champ CSRF correspond*

3.4.1 ...

3.4.2 ...

3.4.3 ...

3.5 Déroulement

3.5.1 Sprints

Sprint 1

User stories planifiée :

[Donner son Token à l'app](#)

[Reprendre mon JDT sur un autre poste](#)

[Changer de Repository et d'utilisateur](#)

[Afficher mes commits sous forme de JDT](#)

Voir [2.4 Analyse fonctionnelle](#) pour plus d'informations.

De ces 4 stories, seulement 2 ont été validée :

[Donner son Token à l'app](#)

[Reprendre mon JDT sur un autre poste](#)

Les 2 autres ont eu de petit bug :

Pour " [Changer de Repository et d'utilisateur](#) " :

Le problème est que pas tous les repositories sont affichés dans la liste.

Le CP a suggéré que cela pourrait être dû à de la pagination du côté de l'API de GitHub.

Pour " [Afficher mes commits sous forme de JDT](#) " :

Pour cette story, tous les tests sont passés excepté 1. Si il y a du Markdown dans le titre ou contenu du commit, il n'est pas mis en page proprement mais simplement affiché comme du text brut.

Les tests de validation ont été effectués sur une machine de l'école qui n'est pas celle où le développement c'est passé. Ils ont également été fait sur ma machine personnelle prise à distance en RDP.

Cela a permis d'avoir un plus large échantillon de machines pour les tests.

J'ai bien avancé sur ce sprint, j'ai été bloqué sur la gestion et affichage du markdown ce qui m'a fait perdre du temps mais à final je ne suis pas en retard.

Pour le sprint 2, je vais commencer par fixer les bugs découverts durant la Sprint Review. Puis suivre la planification initiale.

Sprint 2

User stories planifiées :

[Gérer les entrées](#)

[Exporter en PDF](#)

User stories ayant des bugs lors de la dernière Sprint Review :

Changer de Repository et d'utilisateur

Afficher mes commits sous forme de JDT

Voir [2.4 Analyse fonctionnelle](#) pour plus d'informations.

De ces 4 stories, seulement 3 ont été validées :

[Gérer les entrées](#)

[Exporter en PDF](#)

[Changer de Repository et d'utilisateur](#)

La story " [Afficher mes commits sous forme de JDT](#) " n'a pas été validée dû à une feature non faite, je n'ai pas implémenté l'affichage du Markdown.

Ce sprint a été très productif et j'ai réussi à finir quasiment toutes les stories ainsi que corrigée les stories qui n'avaient pas été validées lors de la SR 1.

Cette sprint review ne se trouve pas à la fin du sprint 2 mais une semaine avant sa fin car le nombre d'heures à ma disposition la semaine prochaine est grandement réduit. Durant les ~13H de la semaine prochaine je compte finir ma doc et si j'ai du temps en plus repasser sur la story non complétée et sur les améliorations rapides. Voir [3.7.3 Dette technique](#) et [3.7.1 Erreurs restantes](#) pour plus d'informations sur ces petits problèmes et features manquante ou avec des aspects insuffisants.

Pour chaque sprint :

- 1. Résumer le déroulement du sprint, citer le résultat (objectif) de sa revue**
- 2. Donner la synthèse de la Sprint retrospective**
- 3. Donner le planning du sprint suivant**

3.5.2 Stories

[Donner son Token à l'app :](#)

Pour cette story, j'ai réutilisé la technique que j'avais utilisé dans mon pré TPI cela m'a donc facilité la vie. Je connaissais déjà les mécanismes et erreurs potentielles ce qui m'a permis de réaliser cette story rapidement et sans perturbations.

Elle a été validée lors de la SR 1.

[Reprendre mon JDT sur un autre poste :](#)

Pour cette story, j'avais pensé et designer la structure du fichier gitj qui est enfaite un fichier json. J'ai donc designer un objet qui match la structure de mon fichier pour la manipulation des données du fichier.

Cela m'a pris un peu de temps à mettre en place au début mais une fois fait, cela m'a fait gagner beaucoup de temps et pas que sur cette story.

J'ai réalisé cette story avant la story qui sauvgarde les données récupérées depuis GitHub donc je me suis servi de l'IA(ChatGPT) pour générer des données bidon pour mes tests.

Cette story m'a pris le temps que j'avais planifié (~10H) et ne m'a pas posé de problèmes

Elle a été validée lors de la SR 1.

[Changer de Repository et d'utilisateur :](#)

Cette storie devait être réalisée lors du sprint 1, ce qui a été fait mais ayant des bugs elle a été validée lors de la sprint review 2.

Lors du sprint 1, tout s'est bien passé jusqu'à la sprint review. Je n'avais pas remarqué mais pas tout mes repositories s'affichaient. Cela était dû à de la pagination comme le CP l'avait suggéré.

Juste après la sprint review 1 ou se problème a été découvert, je me suis mis à fond dessus et l'ai fixé plutôt facilement.

Elle a finalement été validée lors de la SR 2.

[Afficher mes commits sous forme de JDT :](#)

De toutes les User stories, celle-ci est la seule qui n'a pas été validée.

À la base, cette storie était planifiée pour le sprint 1.
Je l'ai réalisée durant le sprint 1, mais sans le markdown car à ce moment j'avais choisi de ne pas le réaliser pour continuer.

Lors de la sprint review 1 tout était ok sauf le markdown.
Pour la sprint review 2, même résultat.

[Gérer les entrées :](#)

Cette storie n'a pas été difficile ou simple, elle s'est déroulée plutôt simplement. Durant cette storie j'ai dû modifier la structure de mon fichier gitj(json) et donc aussi la structure de l'objet qui me sert à stocker ces infos. J'ai rajouté des variables pour savoir quels champs ont été modifiés ce qui me sert à l'affichage (jaune modifié, vert de base).

Elle a été validée lors de la SR 2.

[Exporter en PDF :](#)

Pour cette storie, j'ai changé de librairie en cours de route.
Au départ, j'avais choisi itext7 comme librairie pour la gestion du fichier PDF. Après avoir tester des choses avec je me suis rendu compte que ça allait être trop compliqué de faire ce que je voulais avec. J'ai bifurqué sur PdfSharp qui me paraissait plus simple et effectivement car c'est la librairie que j'utilise.

Hormis ça, je n'ai pas eu de problème sur ce point. Cependant je n'avais jamais fait ça avant ce qui a rendu la tâche plus complexe.

Elle a été validée lors de la SR 2.

3.6 Description des tests effectués

Pour chaque partie testée de votre projet, il faut écrire:

- ~~les conditions exactes de chaque test~~
- ~~les preuves de test (papier ou fichier)~~
- ~~tests sans preuve: fournir au moins une description~~

Faire la synthèse de tous les tests d'acceptance effectués tout au long du projet

3.7 Bilan

3.7.1 Erreurs restantes

S'il reste encore des erreurs :

- Description détaillée
- Conséquences sur l'utilisation du produit
- Actions envisagées ou possibles

3.7.2 Stories

Ce qu'on pensait faire vs ce qu'on a fait

3.7.3 Dette technique

Amélioration possible :

Une fois le JDT display, on peut ajouter des nouvelles entrées, sur la page d'ajout il serait bien de supprimer le champ " Utilisateur " et que ça soit automatiquement l'utilisateur connecté.

Quand on navigue dans le JDT et que l'on modifie les champs, la touche tab ne fait rien. Il faudrait que la touche tab permette de passer de case en case et que shift tab fasse l'inverse.

Lors de l'export du Journal en PDF, une fois le PDF complètement généré il n'y a aucun popup ou tout autre moyen de savoir quand et si le fichier a bien été généré. Il faudrait un moyen de le savoir. Un simple popup comme avec le PAT est suffisant.

Dans le PDF, j'ai fait en sorte que les longues descriptions soient "wrap" c'est-à-dire que des retours sont ajoutés pour éviter que le texte dépasse sur d'autres cases. Par contre les très longs mots ne sont pas pris en compte ce qui certaines fois cause des petits bugs visuels. Il faudrait y remédier.

Reporter la dette technique connue. S'appuyer sur la pratique des // TODO

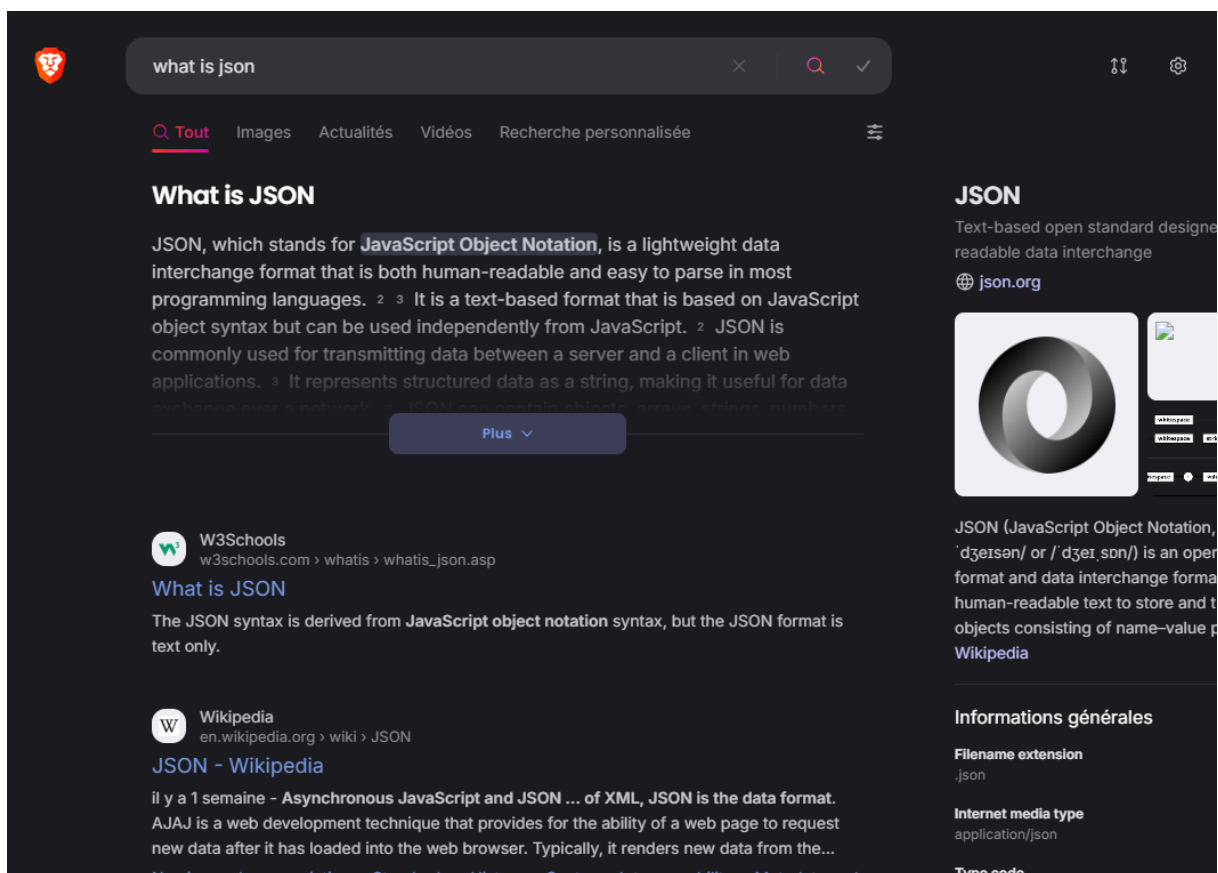
3.8 Recours à l'intelligence artificielle

Durant ce projet j'ai eu recours à l'intelligence artificielle, j'en ai principalement utilisé deux différentes : ChatGPT et Leo AI.

J'utilise Brave comme navigateur et moteur de recherche, c'est là où j'ai eu recours à Leo AI qui est intégrée au moteur de recherche.

Mon utilisation de Leo AI se restreint à du résumé de site web, document, PDF.

Lors d'une recherche sur Brave (moteur de recherche), Leo génère un résumé d'un des premiers sites et ce résumé est placé en haut de la page avant tout lien.

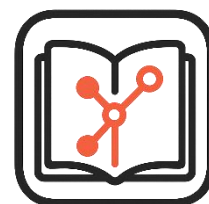


De temps en temps, j'ai consulté ce résumé même si en règle générale je vais chercher de moi-même sur les sites et forums.

J'ai également eu recours à ChatGPT pour plusieurs choses.

Premièrement il m'a permis de créer un logo et une icône simple sans y perdre trop de temps. Ce logo et icône ont également été fait en blanc.

Pour la génération d'image il m'a aussi servi pour mes premières maquettes.



ChatGPT m'a aussi été utile pour accélérer mes recherches. Par exemple pour l'export du JDT en PDF, j'ai utilisé ChatGPT pour me résumer les différents moyens pour arriver à mon but. Malheureusement, ça ne m'a pas empêché de partir sur une certaine librairie et après de multiples bugs et problème de compatibilité, j'ai changé de librairie.

Je l'ai également utilisé pour me générer de fausses données pour des test (Json pour le mettre dans le fichier gitj car à ce moment je n'avais pas encore l'import depuis GitHub).

Mais ou il a été le plus utile est pour les petits bug, quand j'avais un problème que je n'arrivais pas à résoudre ou comment trouver la solution en ligne, je passais pas ChatGPT pour gagner du temp et surtout pour pas en gâcher. Certaine fois quand je n'avais pas beaucoup d'avance, j'utilisais ChatGPT sans faire trop de recherche au préalable.

**Comment avez-vous utilisé l'IA dans votre projet.
Si vous ne l'avez pas utilisée, pourquoi ?
Ce chapitre doit contenir au minimum 200 mots**

3.9 Liste des documents fournis

Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions

- le rapport de projet
- le manuel d'Installation (en annexe)
- le manuel d'Utilisation avec des exemples graphiques (en annexe)
- autres...

4 Conclusions

Développez en tous cas les points suivants :

- Objectifs atteints / non-atteints
- Bilan personnel : points positifs / négatifs
- Difficultés particulières
- Suites possibles pour le projet (évolutions & améliorations)

5 Annexes

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

5.2 Sources – Bibliographie

Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur) ... Et de toutes les aides externes (noms)

5.3 Journal de travail

Date	Durée	Activité	Remarques

Référence à votre journal de travail (en PDF)

5.4 Manuel d'Installation

5.5 Manuel d'Utilisation

5.6 Archives du projet

Media, ... dans une fourre en plastique