
Git
Journal

Table des matières

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 temp car il permet de ne pas perdre de temp 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 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 donné.
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 temp 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 fais la fois d'avant et ce que je compte faire aujourd'hui.

Décrivez comment vous allez gérer votre projet. Quels sont les outils, les pratiques, les personnes impliquées, les rôles,… ?

1.4 Planification initiale

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

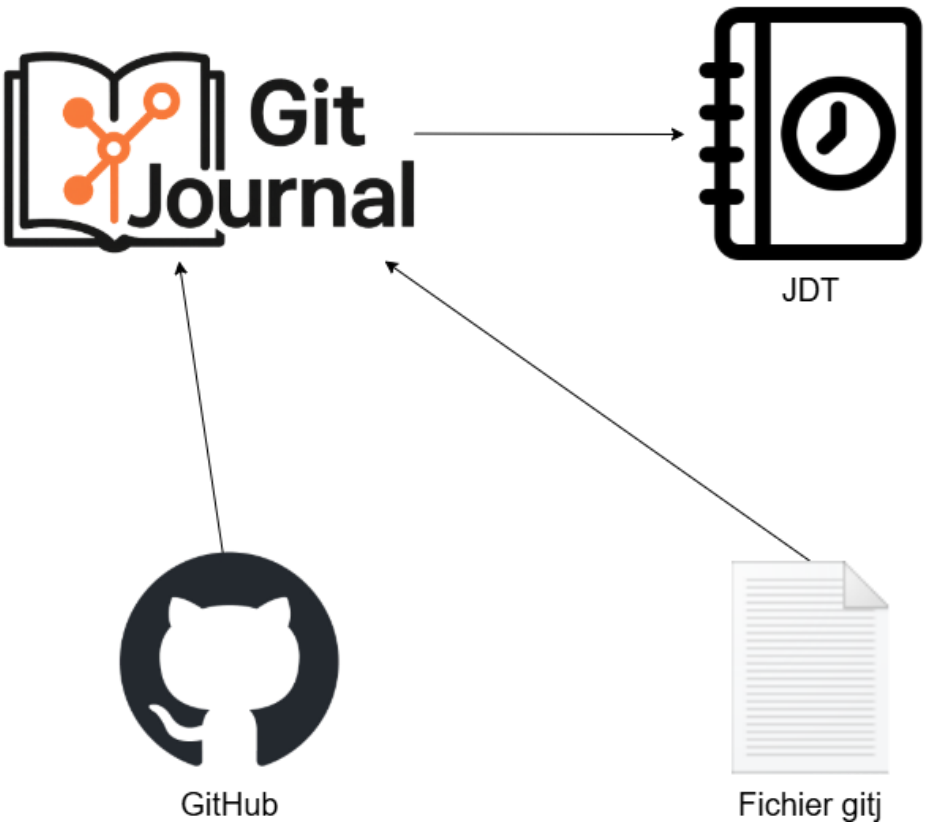
	28.04 - 02.05		05.05 - 09.05		12.05 - 16.05		19.05 - 23.05		26.05 - 30.05		02.06 - 06.06	
	Matin	Après Midi	Matin	Après Midi	Matin	Après Midi	Matin	Après Midi	Matin	Après Midi	Matin	Après Midi
Lundi	-		05:35		05:35		05:35		03:10	Examens	03:10	-
Mardi	-		-		-		-		-		-	
Mercredi	-		07:15		07:15		07:15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

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 métadata des commits GitHub et prend en compte les modifications de l'utilisateur enregistrées dans un fichier gitj.



Placer le produit dans son contexte d'utilisation. Par exemple :

- ***Site Web Internet***
- ***Application, Web, intranet***
- ***Application mobile***
- ***Infrastructure dupliquer des sites multiples***

Présenter les utilisateurs du système ainsi que les rôles qu'ils peuvent avoir
Présenter les données/information que le système prend en charge

Au minimum :

- *Un ou plusieurs schémas de contexte montrant le produit dans son environnement d'utilisation, ainsi que ses utilisateurs. Ce type schéma doit être accompagné d'explications textuelles*

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.
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.
Le code est exécuté directement depuis Visual Studio 2022.

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

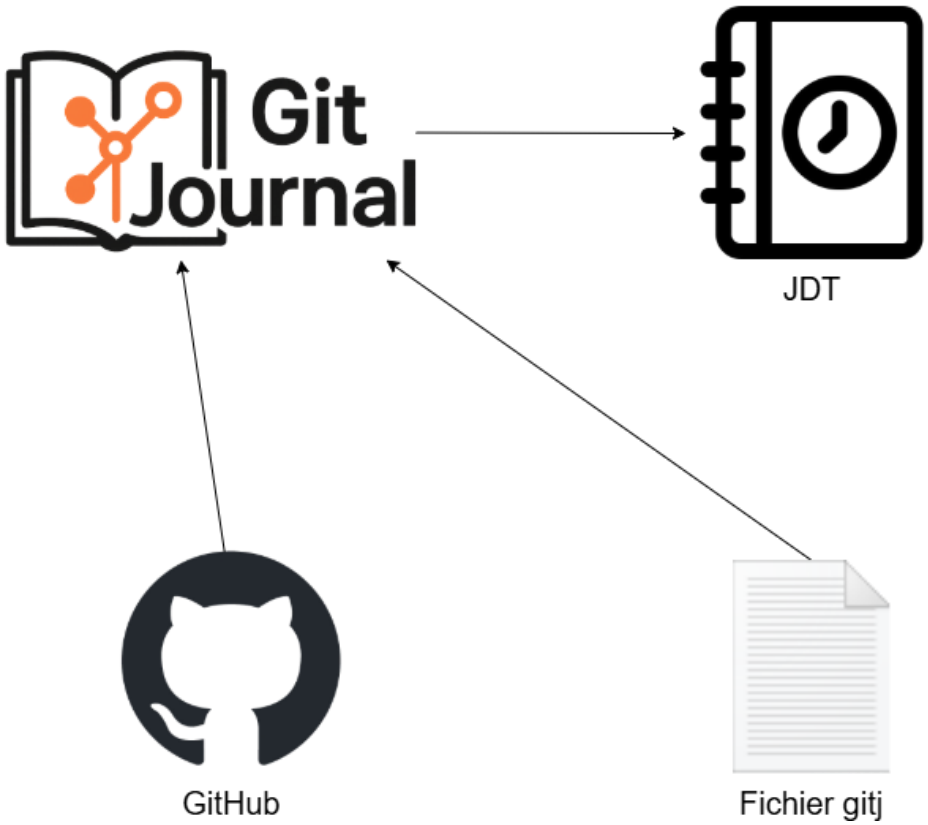
Pour passer de l'environnement de développement à l'environnement de validation, il n'y a pas grand-chose à faire excepté que le repository soit à jour.

Le développement se passera sur Visual Studio 2022.

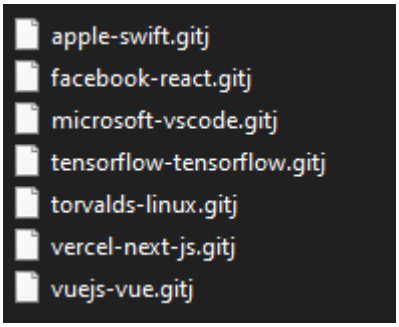
Le programme sera fait en .Net 8, il faudra donc que la machine ait bien .Net 8 d'installé.

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

Décrivez de manière précise en vous appuyant sur des éléments graphiques (maquettes, schémas) la manière dont vous envisagez que votre utilisateur va travailler avec votre produit.

Possibilité (si IceScrum est utilisé) : Reprendre le contenu des User Stories d'IceScrum : Story + tests d'acceptance (avec IceTools) + maquettes

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

2.7 Planification

Révision de la planification initiale du projet :

- ~~*planning indiquant les dates de début et de fin du projet ainsi que le découpage connu des diverses phases.*~~
- ~~*partage des tâches en cas de travail à plusieurs.*~~

~~*Il s'agit en principe de la planification **définitive du projet**. Elle peut être ensuite affinée (découpage des tâches). Si les délais doivent être ensuite modifiés, le responsable de projet doit être avisé, et les raisons doivent être expliquées dans l'historique.*~~

Cette section n'est présente que si la planification initiale a dû être revue suite à l'analyse

3 Réalisation

3.1 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 fromulaire***
  - ***Concatène les noms des indices de $_POST***
  - ***Calcule un hash SHA256 de la chaîne obtenue***
  - ***Vérifie que la valeur du champ CSRF correspond***

3.1.1 …

3.1.2 …

3.1.3 …

3.2 Déroulement

3.2.1 Sprints

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***

3.2.2 Stories

Résumer comment s'est passé la réalisation de chaque story, ses difficultés, les alternatives envisagées mais rejetées, ses surprises, …

3.3 Description des tests effectués

~~*Pour chaque partie testée de votre projet, il faut décrire:*~~

- ~~*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

3.4 Bilan

3.4.1 Erreurs restantes

S'il reste encore des erreurs :

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

3.4.2 Stories

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

3.4.3 Dette technique

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

3.5 Recours à l'intelligence artificielle

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

3.6 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