TPI : Advanced To Do List

Table des matières

[Introduction 3](#_Toc166656904)

[1 Planification initiale 5](#_Toc166656905)

[2 Analyse 9](#_Toc166656906)

[2.1 Méthodologie 9](#_Toc166656907)

[2.1.1 Méthode de gestion de projet 9](#_Toc166656908)

[2.1.2 Sauvegarde 9](#_Toc166656909)

[2.2 Stratégie de test 10](#_Toc166656910)

[3 Réalisation 11](#_Toc166656911)

[4 Tests 12](#_Toc166656912)

[5 Conclusion 13](#_Toc166656913)

[6 Annexes 14](#_Toc166656914)

# Remerciements

Avant d’entamer ce rapport, j’aimerais remercier les gens qui ont été impliqué de loin ou de près dans ce projet.

Tout d’abord, je tiens à remercier mon chef de projet, Monsieur Chenaux, d’avoir accepté d’être mon chef de projet pour ce TPI.

Je remercie également et particulièrement les deux experts, Monsieur Berney et Monsieur Wenger,

# Introduction

Dans le cadre de ma formation d’informaticien et dans le but d’obtenir mon Certificat Fédéral de Capacité d’informaticien, un Travail Personnel Individuel est à réaliser pour conclure mon apprentissage. L’objectif étant de mettre en pratique ce que j’ai appris durant ces deux années passées à l’Ecole des Métiers – Ecole Technique de Lausanne.

Ayant la possibilité de choisir dans quel domaine nous souhaitons faire ce projet, j’ai décidé de me lancer dans un projet de « Développement Desktop » car c’est dans cela que je souhaite me spécialiser par la suite.

*Ce chapitre décrit brièvement le projet, le cadre dans lequel il est réalisé, les raisons de ce choix et ce qu'il peut apporter à l'élève ou à l'école. Il n'est pas nécessaire de rentrer dans les détails (ceux-ci seront abordés plus loin) mais cela doit être aussi clair et complet que possible (idées de solutions). Ce chapitre contient également l'inventaire et la description des travaux qui auraient déjà été effectués pour ce projet.*

# Planification initiale

Ma planification initiale inclut des plages de 1h30 les vendredis, intitulées « Révision globale du code ».

Ces sessions sont initialement prévues pour être dédié à l’optimisation du code, revoir des parties éventuellement obsolètes, … cependant, selon les besoins, ces 1h30 peuvent être également utilisées pour rattraper tout retard dans d’autres aspects du projet, comme la rédaction du rapport.























































# Analyse

## Méthodologie

### Méthode de gestion de projet

La méthode de gestion de projet « Waterfall », plus particulièrement la méthode des « 6 pas » a été choisie et va être appliqué à ce projet.

Le cahier des charges ayant clairement été défini et validé au lancement du projet, la méthode des 6 pas s’est rapidement présentée comme une solution fiable, envisageable et adaptée à la situation.

De plus, étant une personne que l’on peut qualifier de perfectionniste, cette méthode structurée me permettra de m’éviter de tomber dans un peaufinage excessif des détails, qui conduirait à une mauvaise gestion du temps et de prendre un potentiel retard. De ce fait, la progression linéaire et segmentée des 6 pas me servira de fil conducteur et m’encadrera dans mon travail.

Les 6 étapes de cette méthode sont les suivantes :

* **S’informer**
* **Planifier**
* **Décider**
* **Réaliser**
* **Contrôler**
* **Evaluer**

Chaque étape

### Sauvegarde

La perte de données est un risque omniprésent, encore plus dans un projet comme celui-ci et plus généralement dans le métier d’informaticien. Il est crucial de hautement considérer ce danger.

C’est pourquoi j’ai décidé d’établir une « routine » de backup facile à mettre en place afin de minimiser ce risque sans que cela ne soit trop coûteux en termes de temps.

La routine est la suivante :

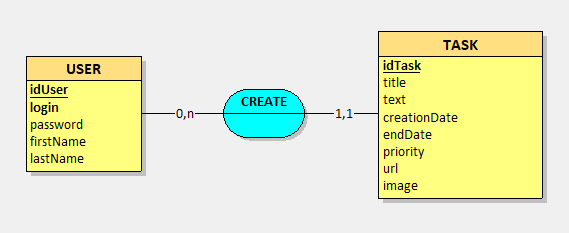
* **Utilisation de GitHub :** GitHub, en plus de permettre le versioning et d’optimiser la portabilité du projet, est une véritable solution de sauvegarde viable. C’est pour ces raisons que j’ai décidé de créer mon projet « actif » sur GitHub.
* **Push régulier sur GitHub :** Au minimum, avant chaque pause ou fin de journée (toutes les 1h30), un commit et un push sont effectués, afin de garantir que les dernières modifications ont été sauvegardées.
* **Sauvegarde sur SSD et en local :** En complément de GitHub, une copie du TPI est également sauvegardée sur mon disque dur externe ainsi que sur l’ordinateur de travail, en local, après chaque mise à jour sur GitHub.

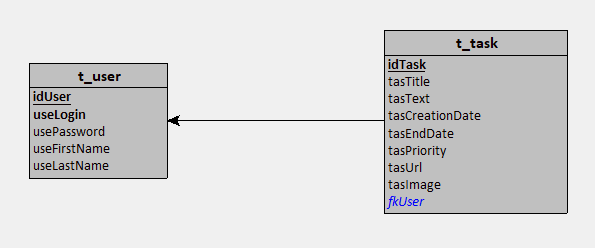
## Stratégie de test

## Conception

### Modélisation de la base de données

Mon application MyTaskManager nécessite une base de données.





# Réalisation

## Compte utilisateur

### Hachage

Les comptes utilisateurs contenant des informations particulièrement sensibles, notamment comme un mot de passe, il est indispensable de protéger et sécuriser ces données.

C’est là que le hachage intervient. En effet, le hachage est un processus unidirectionnel qui permet de convertir une donnée comme un mot de passe en texte chiffré à l’aide « d’algorithmes de hachage »[[1]](#footnote-1).

Dans le cadre de mon application, j’ai opté pour l’utilisation de la bibliothèque BCrypt, principalement car elle offre une méthode de hachage spécialement conçue et adaptée pour les mots de passe. Au-delà du côté « pratique » en termes de développement, BCrypt utilise également le salage dans sa fonction de hachage. Le salage ajoute des caractères aléatoires au hachage, assurant ainsi que chaque hash généré est unique. Par exemple, dans le cas où deux utilisateurs ont le même mot de passe, BCrypt produit deux hashes distincts, ce qui renforce la sécurité et permet d’éviter les « attaques par tables arc-en-ciel »[[2]](#footnote-2).

### Création de compte et login

# Tests

# Conclusion

# Annexes

1. Définition du hachage de Mail in Black :

   https://www.mailinblack.com/ressources/blog/le-salage-de-mots-de-passe-une-couche-de-securite-indispensable/ [↑](#footnote-ref-1)
2. Pourquoi BCrypt est une solution pour le hachage des mots de passe :

   https://nordvpn.com/fr/blog/what-is-bcrypt/ [↑](#footnote-ref-2)