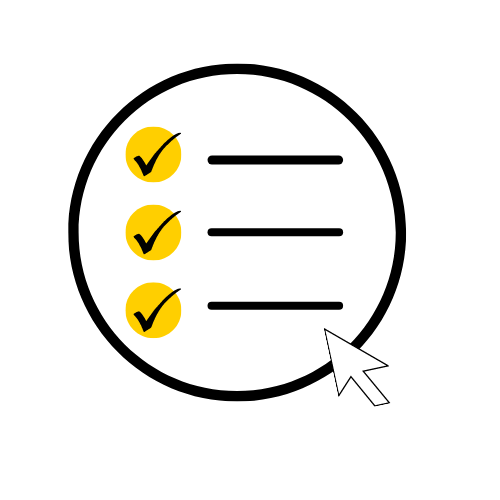
TPI : Advanced To Do List



Théo Ghaemmaghami – FIN2

ETML - Lausanne

Du 13 mai 2024 au 3 juin 2024 – 88 heures

Chef de projet : Monsieur Patrick Chenaux

Table des matières

[Remerciements 4](#_Toc168275190)

[Introduction 5](#_Toc168275191)

[1 Planification initiale 6](#_Toc168275192)

[2 Analyse 7](#_Toc168275193)

[2.1 Méthodologie 7](#_Toc168275194)

[2.1.1 Méthode de gestion de projet 7](#_Toc168275195)

[2.1.2 Sauvegarde 8](#_Toc168275196)

[2.2 Conception 9](#_Toc168275197)

[2.2.1 Modélisation de la base de données 9](#_Toc168275198)

[2.2.2 UwAmp 11](#_Toc168275199)

[2.2.3 Maquettes 11](#_Toc168275200)

[2.2.4 Architecture MVC 15](#_Toc168275201)

[2.3 Stratégie de test 16](#_Toc168275202)

[3 Réalisation 18](#_Toc168275203)

[3.1 Compte utilisateur 18](#_Toc168275204)

[3.1.1 Hachage 18](#_Toc168275205)

[3.1.2 Création de compte 19](#_Toc168275206)

[3.1.3 Login (Connexion à un compte existant) 20](#_Toc168275207)

[3.2 Gestion des tâches 21](#_Toc168275208)

[3.2.1 Objet Tâche 21](#_Toc168275209)

[3.2.2 Ajout d’une tâche 23](#_Toc168275210)

[3.2.3 Affichage des tâches dans les colonnes 25](#_Toc168275211)

[4 Tests 26](#_Toc168275212)

[5 Conclusion 27](#_Toc168275213)

[5.1 Bilan des problèmes rencontrés 27](#_Toc168275214)

[5.1.1 Maladie 27](#_Toc168275215)

[6 Divers 28](#_Toc168275216)

[6.1 Glossaire 28](#_Toc168275217)

[6.2 Table des illustrations 30](#_Toc168275218)

[6.3 Webographie 31](#_Toc168275219)

[7 Annexes 33](#_Toc168275220)

[7.1 Planification initiale 34](#_Toc168275221)

[7.2 Journal de travail 43](#_Toc168275222)

[7.3 Cahier Des Charges 44](#_Toc168275223)

# Remerciements

Avant d’entamer ce rapport, j’aimerais remercier les personnes qui ont été impliquées 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. Dans la même mesure, je souhaiterais le remercier pour sa disponibilité.

Je souhaite également et particulièrement transmettre mes remerciements aux deux experts, Monsieur Berney et Monsieur Wenger, d’avoir accordé de leur temps, d’avoir contribué à l’amélioration de mon projet grâce à leurs retours.

Finalement, je remercie aussi mes camarades de la FIN2 avec qui j’ai pu échanger concernant ce TPI et qui ont participé à la bonification de mon travail.

# 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. En effet, aspirant à poursuivre mes études en Informatique Logicielle à l’HEIG-VD, j’ai vu en ce projet une opportunité de me replonger dans la programmation.

**Présentation de l’application**

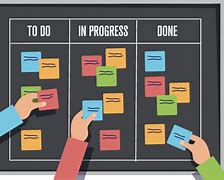
L’application que je suis amené à développer est une « To Do List ». Le programme doit donner la possibilité à l’utilisateur de se créer un compte et de s’y connecter. L’utilisateur doit pouvoir ajouter des tâches avec une certaine quantité d’informations et les gérer à la façon de la méthode Kanban avec des colonnes. La gestion des tâches à travers les colonnes doit se faire via le Glisser-Déposer. Le logiciel doit également offrir un aspect personnalisable : l’utilisateur doit pouvoir ajouter et supprimer des colonnes, renommer le titre de ces dernières et il doit être en mesure de redimensionner la fenêtre comme il le souhaite, sans que cela entrave l’interface et les fonctionnalités de l’application.

Figure 1 : Méthode Kanban

# Planification initiale

Cette planification initiale a été la première étape de ce projet. Elle a été réalisée lors du premier jour de TPI.

Avant tout, je souhaiterais préciser que ma planification initiale inclut des plages de 1h30 les vendredis, intitulées « Révision globale du code ».

Ces sessions sont initialement prévues pour être dédiées 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 appliquée à ce projet.

Les 6 étapes de cette méthode sont les suivantes :

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

Le cahier des charges (par conséquent, les objectifs du projet également) ayant clairement été défini et validé au lancement du projet ainsi que la date d’achèvement, la méthode des 6 pas s’est rapidement présentée comme une solution fiable, envisageable et adaptée à la situation.

De plus, la façon dont sont conçu le TPI et les canevas s’inscrit totalement dans la méthodologie des 6 pas. En effet, la découverte du cahier des charges, les rencontres avec l’expert et le chef de projet au début du projet correspondent à l’étape « S’informer ». La rédaction de la planification initiale est la deuxième étape de ce TPI et s’inscrit dans les étapes « Planifier » et « Décider » de la méthode des 6 pas. La réalisation, comme son nom le démontre explicitement, correspond à l’étape « Réaliser ». L’élaboration de la stratégie des tests et la mise en pratique de ces derniers fait écho à la partie « Contrôler » et « Evaluer ».

En outre, 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 entrainerait 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.

Pour pousser l’utilisation de cette méthodologie un peu plus loin, j’ai décidé de séparer les fonctionnalités demandées comme des « mini-projets » et à chaque mini-projet la méthode des 6 pas est appliquée.

Par exemple, j’ai fait de la « Possibilité de se connecter de manière sécurisée avec des comptes utilisateurs. », qui est une fonctionnalité demandée dans le cahier des charges, un mini-projet.

**S’informer**: Cette étape consiste à comprendre ce qu’il y a à réaliser et prendre conscience des défis à relever. La documentation sur internet peut déjà commencer à ce stade si j’estime que c’est utile.

Qu’est-ce que cette fonctionnalité implique ?

La création d’un formulaire permettant à l’utilisateur de se loguer et un formulaire permettant de créer un compte si l’utilisateur n’a pas de compte.

Y a-t-il potentiellement des points qui vont me poser un ou des problèmes ?

L’aspect « sécurité » de la fonctionnalité (exécution des *requêtes SQL* sécurisée, quelle méthode de hachage du mot de passe sera utilisée, …) et l’aspect « Notification » du modèle MVC (valider qu’une donnée a été insérée correctement dans la base de données et avertir l’utilisateur via la vue).

A partir de là, je commence à me documenter et à effectuer des recherches.

**Planifier et Décider**: Une planification détaillée et initiale ayant déjà été établies, je me fie à ces dernières. Si un choix doit être fait, comme par exemple choisir la fonction de *hachage* (SHA256 ou BCrypt), c’est à ce moment que cela se fait.

Combien de temps maximum j’accorde à la réalisation de cette fonctionnalité (mini-projet) ?

Jusqu’au vendredi 17 mai.

Quels sont les critères de réussite ?

L’utilisateur doit pouvoir créer un compte, se connecter et le processus doit être sécurisé et doit au minimum protéger contre les *injections SQL* et hacher le mot de passe.

**Réaliser** : Cette partie parle d’elle-même. C’est le cœur de la tâche, le moment de programmer. Si de nouveaux problèmes apparaissent durant la réalisation, la documentation et la recherche d’informations peut évidemment également se faire à ce stade.

**Contrôler**: Le contrôle est l’étape de test. C’est le moment de s’assurer que ce qui a été fait dans la partie « Réaliser » répond aux attentes et aux objectifs.

**Evaluer**: Cette partie permet de faire une courte rétrospective de ce qui vient d’être fait. Elle permettra de déterminer plusieurs choses comme des aspects qui auraient été négligés. Si je souhaite y revenir à un autre moment, je commente le code en expliquant ce qui devra être fait ou refait.

Y a-t-il des objectifs supplémentaires que je peux considérer si j’ai davantage de temps à ma disposition ?

La façon dont je traite les « notifications (MVC) », que j’ai laissées de côté, pourrait être revue et optimisée (code répétitif). Le code pourrait globalement être optimisé.

A quel point ces objectifs supplémentaires sont-ils importants ? Vont-ils réellement amener une plus-value au projet ?

L’optimisation du code pourrait rendre le tout plus simple à gérer, mais ce n’est pas primordial car la solution reste fonctionnelle sans cela.

### 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 2 fois par jour. Avant chaque pause de midi et fin de journée, 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.

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 2 : Commits GitHub du début du projet

## Conception

### Modélisation de la base de données

Avant de me lancer dans la création et l’écriture du *script SQL* de ma base de données, j’ai appliqué et suivi la méthode MERISE pour modéliser ma base de données afin de m’assurer de sa cohérence et qu’elle pourra supporter mon application.

MERISE, qui signifie Méthode d’Etude et de Réalisation Informatique pour les Systèmes d’Entreprise, est une méthode d’analyse, de conception et de gestion de projet, surtout utilisée dans le monde de l’informatique.

Dans le cadre de l’élaboration d’une base de données, MERISE propose des « outils » permettant de réaliser une base de données de manière méthodique. Suivre cette méthodologie consiste à d’abord élaborer un Modèle Conceptuel des Données (MCD), qui est un schéma qui met particulièrement en évidence comment les éléments sont liés entre eux, comme le démontre la présence de « *cardinalités* ». Par exemple, dans le schéma ci-dessous, les *cardinalités* indiquent que chaque utilisateur (USER) peut créer zéro ou plusieurs tâches (TASK), alors qu'une tâche doit être créée par un seul utilisateur. Cela souligne une dépendance : une tâche ne peut pas exister sans un utilisateur. Cette observation, nous le verrons plus tard, sera déterminante pour le développement de l’application. En bref, Le MCD est une étape analytique qui permet de faire une première « ébauche » de la structure de la base de données à partir de ce qui est demandé (dans mon cas, par le cahier des charges).

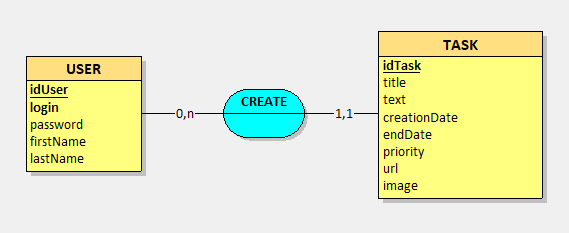


Figure 3 : Modèle Conceptuel des Données

Une fois que le MCD est terminé, le Modèle Logique des Données (MLD) peut être construit. Le MLD reprend le contenu du MCD et le transpose dans un format plus « technique », se rapprochant bien plus du résultat final qui sera prêt à être implémenté dans un Système de Gestion de Base de Données. Cela se reflète notamment dans le nom des *attributs* comme « useLogin » ou encore dans les *entités* TASK et USER qui deviennent des tables et les associations (CREATE), qui représentent les relations entre les entités, qui sont traduites en *clés étrangères*. Tous ces changements seront conservés lors de l’implémentation dans la base de données.

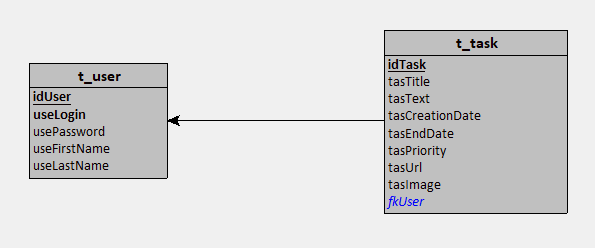


Figure 4 : Modèle Logique des Données (fait sur Looping)

Le Modèle Physique des données (MPD) est la dernière étape de la méthode MERISE dans la conception d’une base de données. Il représente concrètement la structure de la base de données en détaillant notamment les types de données (*int*, *varchar*, …). Contrairement au MCD et au MLD, le MPD est spécifique au système de gestion de base de données, ici *SQL*. Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 5 : Modèle Physique des Données (représentation depuis PHPMyAdmin)

**Explication des spécificités de la table User :**

useLogin est unique. Cela signifie que deux comptes utilisateurs ne peuvent pas posséder le même login. J’ai appliqué cette propriété pour éviter que deux comptes partageant exactement les mêmes informations puissent coexister dans la base de données, afin d’empêcher les brèches de sécurité que cela pourrait provoquer.

**Explication des spécificités de la table Task :**

tasCreation et tasEndDate ont pour type d’attributs « Date » et non pas « DateTime » car je souhaite uniquement stocker des dates, et non pas des dates avec les heures.

tasUrl est un « *blob* » car je souhaite stocker les images directement dans la base de données malgré le fait que cela puisse potentiellement ralentir les performances de cette dernière. Tout d’abord, parce que cette façon de faire est plus simple à gérer. De plus, dans son état actuel, MyTaskManager est un « petit projet ». Stocker directement les images ne risque pas d’avoir de grande conséquence sur les performances et sur l’expérience de l’utilisateur. De plus, comme je vous l’expliquerai dans la pratique, j’ai tenté de limiter l’espace de stockage que prennent les images en portant une attention sur le format de ces dernières.

### UwAmp

UwAmp est un serveur *Wamp* léger et portable permettant d’exécuter Apache, PHP, PHPMyAdmin, etc. qui donne la possibilité de tester son site web ou encore sa base de données.

J’ai choisi d’utiliser UwAmp en lieu et place de Docker car UwAmp est une solution portable, ne nécessitant aucune installation. A travers les divers projets (PHP, MySQL) que j’ai menés, je me suis familiarisé avec UwAmp et toujours dans cette quête de gestion du temps, j’ai préféré privilégier un logiciel que je connais très bien et qui répond à ce dont j’ai besoin pour réussir ce projet à la place de Docker, qui, malgré ses nombreuses qualités, n’est pas un outil avec lequel j’ai eu l’opportunité de travailler.

### Maquettes

Pour établir une cohérence graphique, j’ai réalisé des maquettes de chaque interface de mon application sur Figma.

Le logo, étant un symbole visuel qui représente l’identité d’une marque, d’un logiciel, etc., j’ai, de ce fait, commencé par la conception du logo de mon application. J’ai tenté d’utiliser ChatGPT 4 pour générer un logo correspondant à une application « To Do List », en vain. Les résultats proposés étaient soit trop brouillon ou ils ne me satisfaisaient pas malgré les multiples tentatives, mais cela m’a néanmoins permis de mieux cerner l’identité visuelle que je souhaitais donner à ce projet.



Figure 6 : Post-It

J’ai donc fini par réaliser le logo moi-même sur Canva. Je voulais une image simpliste et parlante. J’ai choisi le jaune comme couleur principale car, au-delà d’être une couleur qui attire l’œil, elle me rappelle les Post-It que beaucoup utilisaient pour noter leurs tâches.

Une image contenant clipart, cercle, diagramme, Graphique

Description générée automatiquement

Figure 7 : Logo de MyTaskManager

Les fondamentaux posés, j’ai pu ensuite me lancer dans l’élaboration des différentes interfaces qui composeraient mon application.

Pour clarifier le fonctionnement de l’interface de mon application, je vais expliquer le lien que les maquettes ont entre elles.

Lorsque l’on démarre l’application, la fenêtre de login est la première à apparaître. A partir de là, on peut tenter de se connecter à un compte en appuyant sur Valider et accéder à l’interface principale si la connexion est réussie ou on peut accéder au formulaire de création d’un compte en cliquant sur le bouton « Créer un compte ».

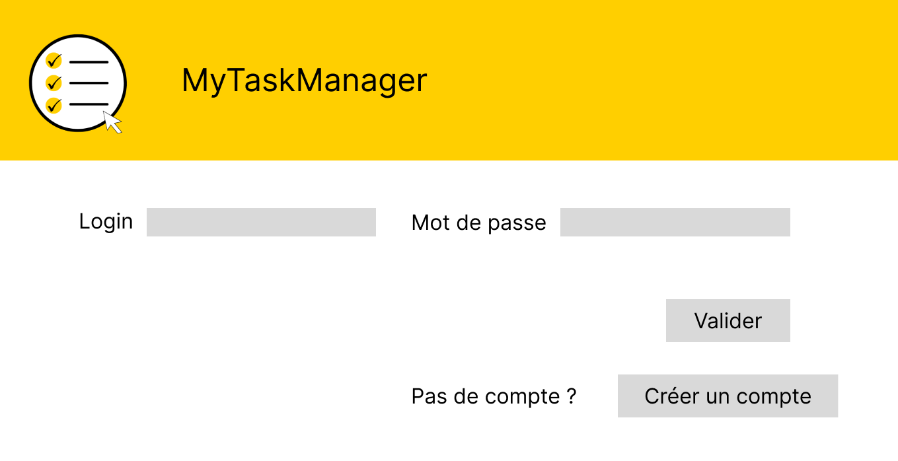


Figure 8 : Maquette de l’interface de connexion

Lorsque l’on valide la création d’un nouveau compte utilisateur, la fenêtre ci-dessous se ferme pour revenir sur l’interface de connexion.

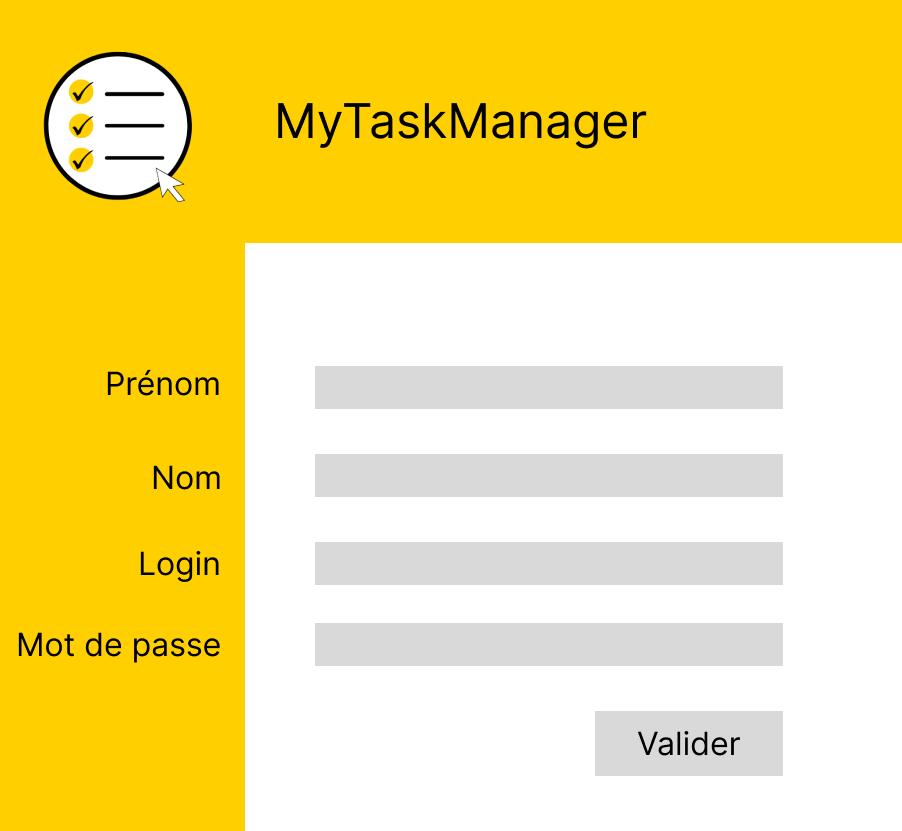


Figure 9 : Maquette de l'interface de création d'un compte

Pour ce qui est de l’interface principale, j’ai rapidement réalisé deux concepts. La version 1 ne me convenait pas visuellement et allait probablement être difficile à gérer, notamment par rapport à l’aspect *personnalisation* de l’application avec la gestion des colonnes. C’est pour ces raisons que mon choix s’est finalement porté sur le deuxième concept.

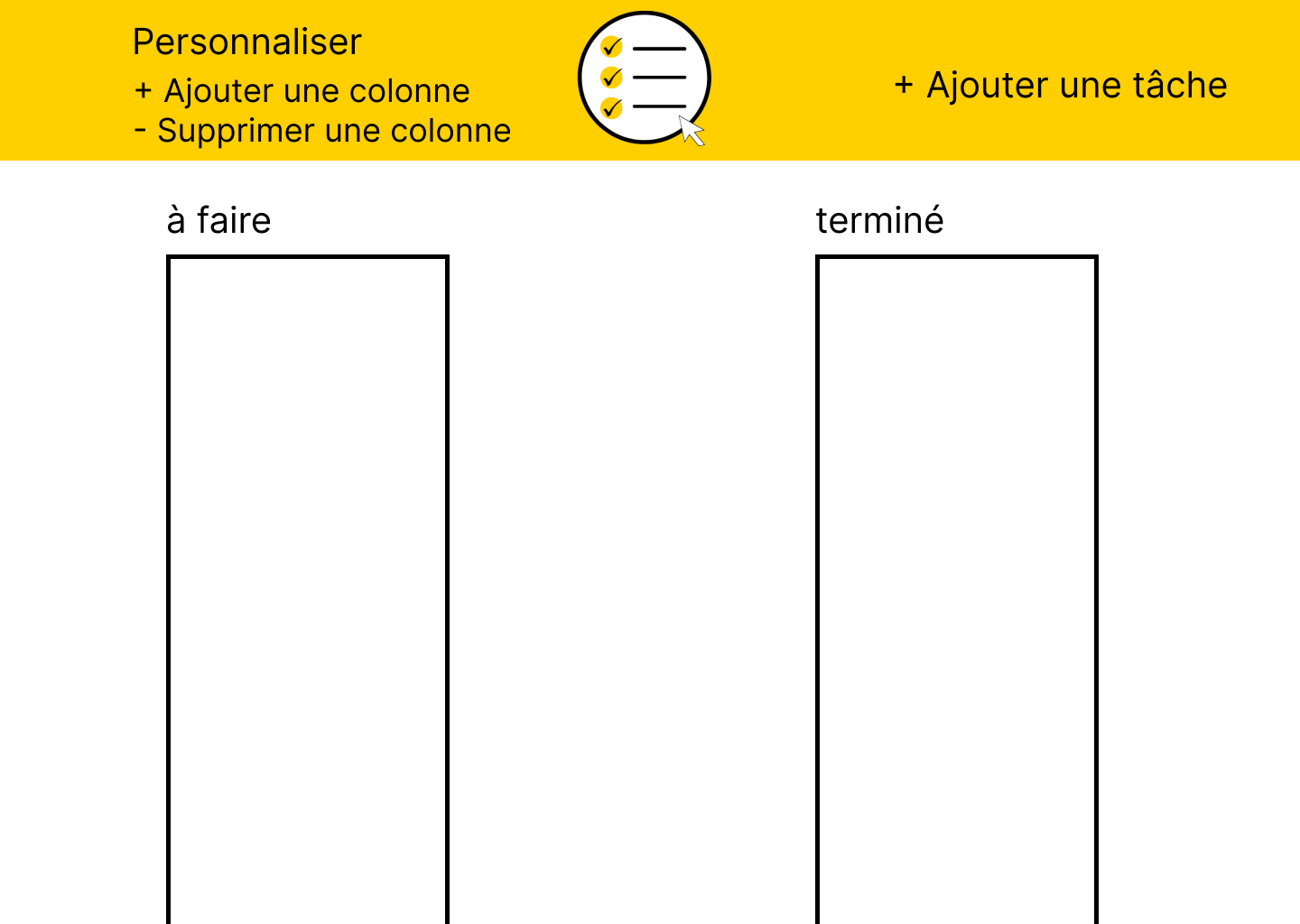


Figure 10 : Maquette de l'interface principale (version 1)

L’interface principale de MyTaskManager permet à l’aide du clic sur le bouton « Ajouter une tâche » d’ouvrir la fenêtre d’ajout d’une nouvelle tâche. Les boutons + et – « Colonne » permettent simplement d’ajouter ou de supprimer une colonne. Ils modifient simplement la fenêtre.

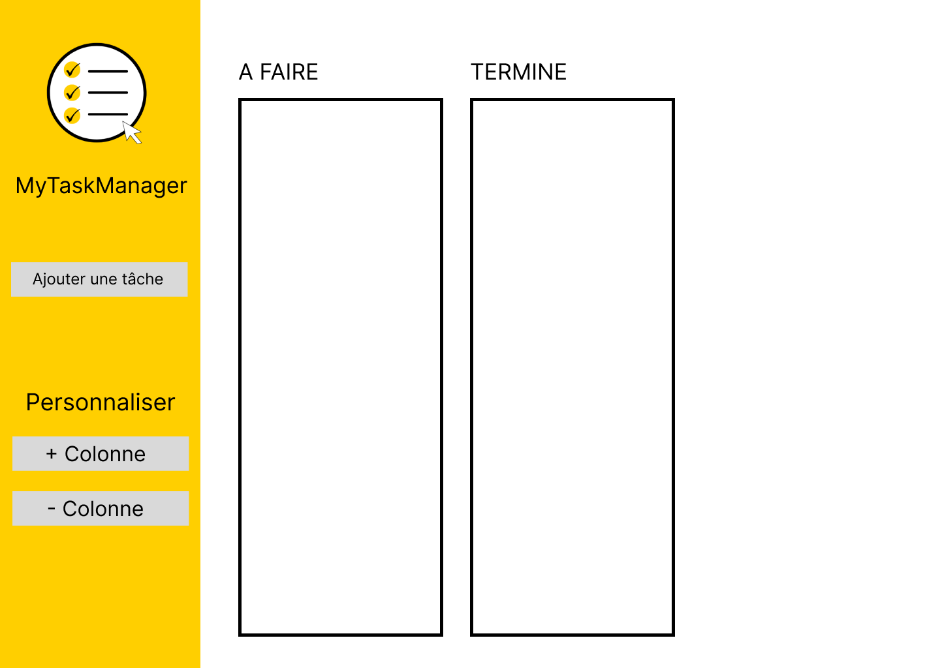


Figure 11 : Maquette de l'interface principale (version finale)

La fenêtre « Ajout d’une nouvelle tâche » s’ouvre par-dessus l’interface principale et lorsque l’utilisateur clique sur le bouton « Valider », la fenêtre se ferme pour revenir sur l’interface principale.



Figure 12 : Maquette de l'interface d'ajout d'une tâche

### Architecture MVC

MVC est une architecture logicielle qui permet de séparer le développement d’une application en 3 « composants » distincts : **Modèle, Vue et Contrôleur.**

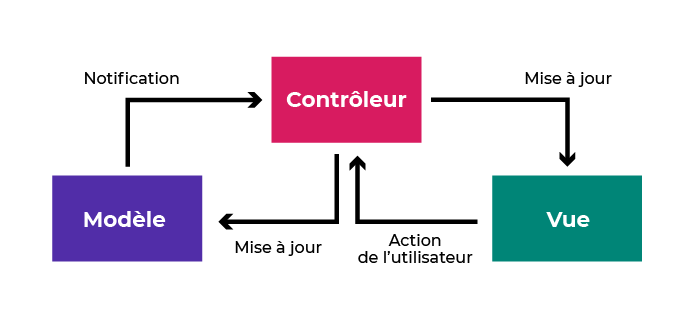


Figure 13 : Fonctionnement du MVC schématisé

**Chaque composant se charge d’un aspect spécifique de l’application.**

**Le Modèle :** Il contient les données manipulées par le programme. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d’une base de données, c’est le modèle qui a accès à la base de données[[1]](#footnote-1). Concrètement, il contient les opérations servant à manipuler les données, en incluant, par exemple, une méthode permettant d’ajouter (créer) un nouvel utilisateur.

**La Vue :** Elle gère toute la logique interface utilisateur. Elle génère l’interface que l’utilisateur va utiliser. Elle est chargée d’afficher les données (fournies par le Modèle) et de recevoir les actions de l’utilisateur (clic de souris, remplissage d’un champ, etc.).

**Le Contrôleur :** C’est l’intermédiaire entre la Vue et le Modèle. Le contrôleur reçoit les événements enclenchés par l’utilisateur depuis la Vue et avertit, par conséquent, le modèle en lui indiquant quoi faire. Inversement, le modèle notifie le contrôleur des changements pour que ce dernier mette à jour la Vue.

Le pattern MVC étant la seule architecture logicielle que l’on a réellement étudiée et dans l’intérêt de « structurer » le code, ce modèle MVC a donc logiquement été utilisé et suivi pour le développement de l’application.

Voici comment j’ai organisé mon MVC :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquementToujours dans l’optique de structurer le code et de répartir spécifiquement les aspects de l’application, j’ai opté pour une séparation entre la partie Utilisateur et la partie Tâche du programme. C’est pourquoi j’ai deux Contrôleurs (MyTaskController et UserController) ainsi que deux Modèles (MyTaskModel et UserModel).

De plus, il faut noter la présence de l’*objet* Colonne dans les Vues. Cet *objet* sert uniquement à l’interface principale de l’application, permettant à l’utilisateur de gérer les tâches. Les tâches s’affichant dans les colonnes, j’ai spécialement donné l’accès à *l’objet* Colonne à *l’objet* Tâche (MyTask) malgré le fait que cela ne respecte pas le principe de l’architecture MVC. Colonne étant un *objet* uniquement visuel, cela n’aura pas d’impact sur l’implémentation du pattern MVC dans le reste du programme.

Figure 14 : MVC de MyTaskManager

## Stratégie de test

Pour garantir le succès d’un projet, il est fondamental de s’assurer que le résultat du travail accompli soit fonctionnel. Pour ce faire, il faut réaliser des tests. Ces tests permettent de tester les limites de la solution et de détecter d’éventuelles erreurs. Tester est laborieux car il y a de nombreux types de test qui existent et chaque projet nécessite une façon de tester spécifique. C’est pourquoi il est important de planifier une stratégie de tests efficace et adaptée, permettant de couvrir l’étendue de l’ampleur du projet.

Dans le cadre du développement d’une application fonctionnant avec une base de données *SQL*, il est important de contrôler trois éléments majeurs : fonctionnement des fonctionnalités, sécurité, optimisation de l’expérience utilisateur.

Les fonctionnalités demandées par le cahier des charges doivent être fonctionnelles et répondre à un certain niveau d’exigeance.

La sécurité doit être primordiale car des données privées vont être potentiellement stockées dans la base de données.

L’expérience utilisateur ne doit pas être entravée par des *bugs* ou par manque de facilité d’utilisation.

Pour définir clairement l’objectif de mes tests, j’ai décidé d’utiliser le principe des « User Stories ». Un récit utilisateur ou User Story est une phrase servant à expliquer une fonctionnalité logicielle, rédigée du point de vue de l’utilisateur final[[2]](#footnote-2).

Après avoir écrit la User Story qui sera l’objectif du test, je passe à l’élaboration d’un scénario de test correspondant. Je définirai ensuite le résultat attendu.

L’étape suivante consiste à exécuter le test conformément au scénario établi. Une fois le test terminé et le résultat obtenu, je procède à sa validation. Si le test est concluant, je le valide et documente les résultats. Si une correction est réalisée, je la mentionnerais et l’expliquerais à condition qu’elle permette un résultat positif à la réexécution du test.

# Réalisation

Le chapitre « Réalisation » va couvrir toute la partie « Pratique » de ce projet. A travers ce chapitre, je vais expliquer le fonctionnement du code de mon application, raconter et justifier mes choix pour vous donner un aperçu de ma façon de faire et de penser.

J’aimerais d’abord clarifier la chronologie de la réalisation. L’application MyTaskManager peut être séparée en deux parties : gestion des utilisateurs et gestion des tâches.

Nous avions constaté durant la partie *Analyse*, lors du chapitre [*Conception de la base de données*](#_Modélisation_de_la), qu’il y avait une dépendance : une tâche ne peut pas exister sans un utilisateur. De ce fait, je me suis d’abord occupé de programmer la partie *Utilisateur* de l’application. La gestion des tâches se fera uniquement lorsque les fonctionnalités liées à la gestion des utilisateurs seront fonctionnelles.

De manière plus générale et à chaque implémentation d’une nouvelle fonctionnalité, j’ai constamment procédé de la même façon : je commence par la Vue (faire l’interface s’il y en a une), je continue avec le Contrôleur et je finis avec le Modèle. C’est pourquoi je vais structurer de la même manière les paragraphes explicatifs de chaque fonctionnalité.

## 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* »[[3]](#footnote-3).

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. BCrypt est pratique en termes de développement car elle offre une *méthode* de *hachage*. Il suffit d’appeler cette *méthode* préconçue et le *hachage* est fait, sans que j’aie eu besoin de me charger de la création de ma propre *méthode*. Au-delà de l’aspect « pratique », 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* »[[4]](#footnote-4).



Figure 15 : Méthode de hachage BCrypt

### Création de compte

Comme je l’ai déjà mentionné précédemment, mon application propose la possibilité de créer un compte utilisateur.

Lorsque l’utilisateur a rempli le formulaire ([design du formulaire](#_Maquettes)) de création d’un compte, au clic du bouton « Valider », l’action est récupérée par la vue et les informations des champs du formulaire sont envoyées au Contrôleur des utilisateurs.

Une image contenant texte, logiciel, Logiciel multimédia, capture d’écran

Description générée automatiquement

Figure 16 : Evénement au clic du bouton « Créer un compte »

Le Contrôleur crée alors un objet Utilisateur à partir des informations récoltées pour l’envoyer au Modèle des utilisateurs.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure 17 : Méthode « Créer un compte » dans le Contrôleur

Le Modèle se charge d’ouvrir la connexion entre l’application et la base de données et, après avoir hashé le mot de passe comme mentionné dans le chapitre précédent, de préparer la *requête SQL* que la base de données va exécuter pour ajouter le nouvel utilisateur.

Afin de protéger mon application et ma base de données des attaques par *injections SQL*, les valeurs ne sont pas directement injectées dans la *requête SQL*. A la place, j’ai utilisé des paramètres SQL (@useFirstName). J’appelle ensuite la *méthode* « AddWithValue » qui permet de lier le paramètre SQL avec la valeur correspondante. Cette étape renforce la sécurité de l’application.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 18 : Méthode « Créer un compte » dans le Modèle

Pour notifier l’utilisateur du résultat du processus, chaque méthode dans le Contrôleur ou dans le Modèle retourne un *tuple* contenant un *booléen* et une exception. Le *booléen* définit si l’opération s’est correctement déroulée. Dans le cas contraire, l’erreur est retournée afin de pouvoir l’afficher dans la Vue.

### Login (Connexion à un compte existant)

Parce que la fonctionnalité de création de compte ne va pas sans la fonctionnalité de connexion à un compte, j’ai implémenté un login.

Après avoir reproduit [la maquette du design du login](#_Maquettes) via le designer et la boite à outils de Visual Studio, j’ai défini l’événement Clic lorsque l’utilisateur clique sur le bouton *Connexion*. La Vue envoie les informations entrées par l’utilisateur dans le formulaire au Contrôleur des utilisateurs.

Le Contrôleur fait simplement acte d’intermédiaire et transmet les informations au Modèle des utilisateurs.

La *méthode* LoginUserToDB de mon Modèle agit de la même manière que lors de la création de compte : il ouvre la connexion à la base de données, prépare la *requête SQL* en la protégeant des *injections SQL* via la liaison des données (Data Binding) et l’exécute. A partir de là, il y a des changements notables. La *requête SQL* est non-seulement exécutée mais également lue par un objet « MySqlDataReader » qui permet, comme son nom le suggestionne, de lire les données récupérées par l’exécution de la *requête SQL*. Si une ligne est lue avec le login entré par l’utilisateur, cela signifie que le compte utilisateur, auquel l’utilisateur actuel essaie de se connecter, existe dans la base de données. Alors, le mot de passe haché est récupéré et comparé au mot de passe inséré par l’utilisateur via une *méthode* de la *bibliothèque* BCrypt. Si les mots de passe correspondent, l’ID du compte utilisateur est retourné.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 19 : Méthode BCrypt.Verify()

De nouveau, afin de notifier l’utilisateur du résultat du processus, la méthode du Modèle et du Contrôleur retourne un tuple contenant un entier *nullable* et une exception. L’entier représente l'ID de l'utilisateur s'il a réussi à se connecter, sinon il est *null*. En cas d'erreur, l'exception est retournée afin de pouvoir l’afficher dans la Vue.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 20 : Gestion des erreurs lors du login (dans la Vue)

Dans le cas où l’erreur est un mot de passe incorrect, l’utilisateur est informé. En effet, si aucune exception n’est retournée (*null* car il n’y a pas eu d’erreur au niveau du Modèle) et que l’ID de l’utilisateur est également *null*, cela signifie que le problème provient de l’utilisateur qui n’a pas entré le bon mot de passe. Cela permet de faire la distinction entre les erreurs liées aux identifiants de connexion incorrects et les autres erreurs potentielles.

## Gestion des tâches

Maintenant que la création de compte utilisateur et l’authentification aux comptes existants sont fonctionnelles, il est temps de s’occuper de la partie « Tâche » de l’application.

### Objet Tâche

MyTaskManager étant un programme qui doit permettre à l’utilisateur de pouvoir générer et gérer des tâches, j’ai créé un objet MyTask, qui est *hérité* de *Panel* (pour l’aspect visuel). Pour connaître les *attributs* et les *paramètres* de ma classe MyTask, il faut se référer au chapitre [Modélisation de la base de données](#_Modélisation_de_la).

Il faut préciser que *l’attribut* et *paramètre* « Image » d’une tâche est un tableau d’octets car, puisque les tâches sont stockées dans une base de données, *l’attribut* image (dans ma base de données) étant de type *BLOB*, il est nécessaire de convertir l’image en octets pour pouvoir la stocker.

Une image contenant texte, Police, jaune, capture d’écran

Description générée automatiquementUne image contenant texte, jaune, capture d’écran, Police

Description générée automatiquementLa *méthode* «DisplayTask »est la *méthode* qui est *appelée* dans l’interface principale pour afficher visuellement la tâche. Elle configure l’apparence, affiche et place les *attributs* dynamiquement.

Figure 21 : Variante de l'apparence d'une tâche

Figure 22 : Apparence d'une tâche dans l'application

Comme il est possible de le constater, les *attributs* d’une tâche apparaissent de manière dynamique selon les champs qu’a renseigné l’utilisateur. En effet, excepté pour le titre qui est obligatoire, les autres éléments sont facultatifs. J’effectue donc une vérification pour chaque élément via un *if()*.

La suite de ma classe MyTask est composée de *méthodes* qui permettent de modifier l'apparence de la tâche en fonction des actions de l'utilisateur, comme par exemple un changement de couleur de la tâche lorsque l’utilisateur passe son curseur par-dessus. J’ai mis cela en place pour montrer qu’une interaction est possible, dans ce cas pour indiquer à l’utilisateur que la tâche est déplaçable (Glisser-Déposer).

Logiquement, je possède également une *méthode* permettant le déplacement des tâches.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Figure 23 : Méthode MyTask\_MouseDown qui permet de rendre les tâches déplaçables

MouseDown est un événement qui se produit lorsque l’utilisateur a son curseur sur le contrôle et un bouton de la souris enfoncé (ici le clic gauche). Au moment où cette action se produit, la donnée sur laquelle le clic est enfoncé (ici la tâche), cette dernière est déplacée.

Pour finir, j’ai également une *méthode* « LblUrl\_Click() »qui permet d’ouvrir et d’accéder au lien contenu dans le lien *URL*.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 24 : Méthode permettant d'ouvrir l'URL contenue dans la tâche

### Ajout d’une tâche

Une fois que l’utilisateur est connecté, il accède à l’interface principale de MyTaskManager. De là, il peut accéder au formulaire de création d’une tâche, [dont le design se trouve ici](#_Maquettes).

Une tâche peut contenir une image. Pour ce faire, j’ai mis en place un *panel* dans lequel l’utilisateur peut glisser-déposer une image pour l’enregistrer.

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

Figure 25 : Glisser-Déposer de l'image de la tâche

*\_imgData* est un tableau d’octets qui *appelle* une *méthode* permettant de convertir l’image, que l’on a glissé-déposé dans l’application, en tableau d’octets pour, comme je l’avais mentionné dans le chapitre précédent *Objet Tâche et affichage,* pouvoir la stocker dans la base de données.

*Une image contenant texte, capture d’écran, Police

Description générée automatiquement*

Figure 26 : Méthode pour convertir une image en tableau d'octets

Après avoir rempli les informations du formulaire, l’utilisateur valide en appuyant sur le bouton « Ajouter une tâche ». Après avoir vérifié que le champ titre n’est pas vide (car pour que la création d’une tâche soit valide, seul le titre est obligatoire), les informations sont envoyées au Contrôleur des tâches.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 27 : Evénement au clic du bouton "Ajout d'une tâche"

Il faut noter que l’événement « TaskAdded » est déclenché. Nous verrons son utilité lorsque nous parlerons de l’affichage des tâches dans les colonnes.

Le Contrôleur des tâches définit l’heure de création de la tâche et créé un objet Tâche à partir des informations reçues de la Vue, pour l’envoyer au Modèle.

La *méthode* « AddTaskToDatabase » du Modèle des tâches est similaire en tout point à [la *méthode* permettant d’enregistrer un nouveau compte utilisateur](#_Création_de_compte).

### Affichage des tâches dans les colonnes

La colonne est un objet visuel qui hérite de Panel dans lequel s’afficheront les tâches.

# 

# Tests

Pour ce qui est des résultats obtenus :

- OK, signifie que tout s’est déroulé correctement selon le scénario de test et le que le résultat attendu a été obtenu

- KO1, signifie que la première étape du scénario de test ne s’est pas déroulé comme prévu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objectif du test** | **Scénario de test** | **Résultat attendu** | **Résultat obtenu** | **Modification(s)** |
| En tant que nouvel utilisateur, je souhaite créer un compte de manière sécurisée. | 1. Accéder au formulaire de création d’un compte 2. Remplir tous les champs avec des informations réalistes 3. Cliquer sur le bouton de validation 4. Affichage de pop-up de validation qui confirme la création | Le compte utilisateur apparaît dans la base de données avec son mot de passe haché | OK |  |
| En tant qu’utilisateur possédant déjà un compte, je souhaite me connecter à mon compte | 1. Accéder au formulaire de connexion (normalement 1ère interface à apparaître) 2. Remplir les champs avec des informations correctes 3. Cliquer sur le bouton de validation 4. Affichage du pop-up de validation qui confirme la connexion 5. Ouverture de l’interface principale 6. Création d’une nouvelle tâche | L’utilisateur a accès à l’interface principale et peut créer d’une nouvelle tâche. FkUser de la tâche créée correspond à celui de l’utilisateur connecté. | OK |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objectif du test** | **Scénario de test** | **Résultat attendu** | **Résultat obtenu** | **Modification(s)** |
| En tant qu’utilisateur possédant déjà un compte mais ne se souvenant plus de ses identifiants, je souhaite me connecter à mon compte. | 1. Accéder au formulaire de connexion 2. Remplir les champs avec le mot de passe erroné mais le login correct 3. Cliquer sur le bouton de validation 4. Affichage du pop-up de validation qui rejette la connexion 5. Réitération avec cette fois le login erroné mais le mot de passe correct | Les deux fois, l’utilisateur doit avoir un pop-up d’erreur et doit être bloqué sur la fenêtre de connexion | OK |  |
| En tant qu’utilisateur connecté, je souhaite ajouter une tâche et la voir apparaître | 1. Accéder au formulaire d’ajout d’une nouvelle tâche depuis l’interface principale 2. Remplir les champs sauf le titre 3. Cliquer sur le bouton de validation 4. Affichage de l’erreur « Titre manquant » 5. Réitération avec un titre 6. Affichage du pop-up de validation d’ajout d’une tâche 7. Fermeture de la fenêtre Ajout d’une tâche 8. Affichage de la tâche dans l’interface principale dans la colonne « A faire » | La tâche doit apparaître dans la colonne A FAIRE et dans la base de données. Les informations doivent correspondre. L’ID de l’utilisateur doit correspondre à la FkUser de la tâche. | OK |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objectif du test** | **Scénario de test** | **Résultat attendu** | **Résultat obtenu** | **Modification(s)** |
| En tant qu’utilisateur connecté venant de terminer des tâches que j’ai ajouté, j’aimerais la déplacer dans la colonne TERMINE | 1. Accéder à l’interface principale 2. Passer le curseur sur une tâche 3. Affichage du changement de couleur de la tâche 4. Sortir le curseur de la tâche 5. Affichage de la couleur par défaut de la tâche 6. Le curseur sur la tâche, enfoncer le clic gauche 7. Déplacer dans la même colonne 8. Affichage du pop-up d’erreur 9. Réitération et déplacement dans une autre colonne | La tâche change de couleur selon si le curseur est sur elle ou pas. La tâche est déplaçable uniquement dans une autre colonne qu’où elle se trouve. | OK |  |
| En tant qu’utilisateur souhaitant personnaliser son interface, je souhaite ajouter une colonne | 1. Accéder à l’interface principale 2. Cliquer sur le bouton Ajouter une colonne 3. Affichage de la nouvelle colonne 4. Glisser-Déposer une tâche dedans 5. Réitération de l’opération jusqu’à 7 colonnes 6. Affichage de la pop-up d’erreur | La colonne apparaît à côté de la colonne TERMINE avec le titre TITRE. La colonne peut accueillir une tâche.  L’ajout est bloqué lorqu’il y a 6 colonnes | OK |  |
| En tant qu’utilisateur souhaitant personnaliser son interface, je souhaite supprimer une colonne | 1. Accéder à l’interface principale 2. Cliquer sur le bouton Supprimer une colonne 3. Suppression de la colonne 4. Réitération de l’opération jusqu’à n’avoir qu’une colonne 5. Affichage de pop-up d’erreur | La dernière colonne (la plus à droite) disparaît. La suppression est bloquée lorsqu’il reste 2 colonnes. | OK |  |
| **Objectif du test** | **Scénario de test** | **Résultat attendu** | **Résultat obtenu** | **Modification(s)** |
| En tant qu’utilisateur souhaitant personnaliser son interface, je peux redimensionner à ma guise sans gêner le contenu | 1. Accéder à l’interface principale 2. Agrandir la fenêtre 3. Ajouter une colonne 4. Diminuer la fenêtre 5. Ajouter une colonne 6. Déplacer une tâche dans la dernière colonne | Les colonnes sont accessibles peu importe la taille de la fenêtre. Le contenu s’affiche correctement | (ok) : les fonctionnalités de l’application sont fonctionnelles mais l’affichage peut présenter des bugs selon le redimensionnement | Manque de temps à investir pour corriger les bugs d’affichage |

# 

# Conclusion

## Bilan des fonctionnalités demandées

## Bilan des problèmes rencontrés

### Maladie

Durant la dernière semaine du TPI, j’ai été malade. Je souligne évidemment le fait que je ne cherche pas à attribuer mon retard à un problème de santé. A ce stade du projet, j’avais déjà un retard dans son avancement. Cependant, mon état maladif m’a ralenti durant trois jours à un certain degré dans ma progression, suffisamment pour que je le mentionne dans ce bilan des problèmes rencontrés.

### Gestion des vues (respect du pattern MVC)

Durant le module i120, nous avons découvert et appris à travailler avec l’architecture MVC. Suivre ce pattern s’est présenté comme une décision naturelle et logique pour la conception de MyTaskManager.

Rapidement, j’ai constaté des lacunes, notamment au niveau de la gestion des vues. En effet, en i120, nous avions conçu des applications pouvant être considérées comme simples, car comportant une vue unique, un seul contrôleur et un modèle. Passer à un programme nécessitant la création de plusieurs interfaces a été un défi, malgré le fait que j’aie pu anticiper ces lacunes lors du projet préparatoire au TPI : P\_Appro2.

Il est à noter que travailler avec deux contrôleurs et deux modèles, bien que visant à séparer les logiques métiers, a ajouté une couche de complexité. Il fallait non seulement attribuer l’accès du bon contrôleur aux bonnes vues, mais il fallait également gérer les problèmes de dépendances lors des instanciations des composants.

Initialement, j’ai tenté d’instancier tous les composants dans le « Main » du programme, mais cela s’est avéré être une mauvaise pratique, notamment à cause des dépendances. J’ai également tenté de créer un gestionnaire de vues à l’aide de ChatGPT mais cette tâche s’est rapidement révélée comme étant peu adaptée et trop complexe à la situation.

Pour résoudre ce problème, j’ai dû revoir ma manière de respecter le pattern MVC et ma façon d’ouvrir les vues. Un contrôleur ayant besoin de base d’une vue et d’un modèle, n’avait besoin plus que du modèle. Les vues sont désormais instanciées directement lors du clic d’un bouton et les vues se donnent l’accès aux contrôleurs entre elles.

## Bilan personnel

A titre personnel et sur un ton un peu plus léger, ce projet m’a permis de me plonger dans la peau d’un développeur et m’a fait comprendre quelque chose : je déteste autant que j’aime la programmation. Ce TPI m’a rappelé à quel point le développement desktop est un domaine où je suis novice.

Le sentiment dominant est celui de la frustration : de ne pas avoir pu peaufiner le programme aussi loin que je l’aurais souhaité, de buter sur des tâches qui, à premiers abords, semblaient simples. Néanmoins, je suis fier de travail accompli et de l’investissement dont j’ai fait preuve, même en dehors des périodes de travail et malgré mon état maladif durant le tiers du projet. Je suis assez satisfait de la qualité du rapport, bien que je regrette de ne pas avoir établi une stratégie de tests plus élaborée et globalement, ne pas avoir consacré plus de temps aux tests, notamment pour faire les tests unitaires.

Concernant la méthodologie de gestion de projet, je reconnais avoir perdu du temps sur des détails qui, cumulés, ont eu un impact significatif sur l’avancement, ce qui m’a contraint à travailler le dernier week-end avant la date du rendu. Si le projet était à refaire, je m’efforcerais de respecter davantage la méthodologie et d'établir une planification plus adaptée. Les tests, notamment les tests unitaires, ne seraient pas négligés.

Quant aux suites à donner au projet, étant donné que je n’ai pas pu atteindre le degré de finition que j’aurais aimé atteindre comme je l’ai expliqué, j’estime que le programme peut être amélioré sur tous les aspects, en particulier en ce qui concerne le redimensionnement et l’adaptation du contenu. La mise en place d’un système CRUD pour les tâches, permettant de les modifier et de les supprimer ainsi que l’amélioration du système de suppression des colonnes, sont des pistes à explorer. Bien sûr, la correction des bugs est une priorité absolue. Même si selon moi la liste des suites à donner à ce travail semblent sans fin, je pense qu’il y a déjà suffisamment à faire.

Ce projet m’a donné envie de m’investir davantage dans la programmation afin de ne plus me sentir novice et m’a incité à explorer de nouveaux frameworks, comme Avalonia, car Windows Forms n’est pas la solution la plus optimale.

# Divers

## Glossaire

A

*appelée*

En programmation. Action d'exécuter une fonction ou une méthode dans un programme informatique en utilisant son nom et, éventuellement, des paramètres, afin d'effectuer une tâche spécifique ou d'obtenir un résultat précis 22

*attaques par tables arc-en-ciel*

Méthode utilisée pour deviner des mots de passe en comparant leurs "empreintes digitales" avec une liste préétablie de mots de passe et leurs empreintes 18

*attributs*

Traits ou caractéristiques décrivant une entité dans une base de données, tels que le nom, l'âge ou l'adresse 10

B

*backup*

Copie de sécurité des données importantes, généralement stockée dans un emplacement sûr pour la récupération en cas de perte ou de dommage des données d'origine 8

*bibliothèque*

Ensemble de ressources prédéfinies et réutilisables, telles que des fonctions, des classes ou des modules, conçues pour faciliter le développement de logiciels en fournissant des fonctionnalités déjà implémentées 18

*blob*

Abréviation de "Binary Large Object" en informatique, représentant un type de données utilisé pour stocker des données binaires, telles que des images, des fichiers audio ou vidéo, dans une base de données 11

booléen

type de variable à deux états "true" ou "false" 20

*bugs*

Problèmes ou défauts dans un logiciel qui provoquent un comportement indésirable ou des erreurs lors de son exécution 17

C

*cardinalités*

Nombre d'éléments différents dans un groupe de données ou le nombre de fois qu'une chose se produit dans une liste 9

*clés étrangères*

Colonnes dans une table de base de données qui établissent une relation avec une clé primaire dans une autre table, permettant ainsi de lier les données entre les différentes tables 10

E

*entités*

Objet ou chose identifiable dans une base de données, comme une personne, un produit ou un lieu 10

H

*hachage*

Conversion de données en une série de caractères aléatoires, souvent utilisée pour sécuriser des informations 8

*hérité*

Terme utilisé en programmation orientée objet pour décrire le processus par lequel une classe (ou objet) peut acquérir les propriétés et les comportements d'une autre classe (ou objet) appelée classe parente 21

I

*if()*

Mot-clé utilisé en programmation pour évaluer une condition et exécuter un bloc de code spécifique si cette condition est vraie 22

*injections SQL*

Méthode utilisée par les pirates informatiques pour insérer du code malveillant dans des requêtes SQL, souvent dans le but de compromettre une base de données ou de récupérer des informations sensibles 8

*int*

Abréviation de "entier" en informatique, représentant un type de données qui stocke des nombres entiers (sans décimales) 10

M

*méthode*

Bloc de code regroupant des instructions pour effectuer une tâche spécifique, associée à un objet ou à une classe dans la programmation orientée objet 18

N

*nullable*

Capable de contenir une valeur null (absence de valeur) 21

O

*objet*

En informatique. Structure de données contenant à la fois des données et des méthodes qui opèrent sur ces données, permettant ainsi de modéliser des entités ou des concepts du monde réel de manière abstraite dans un programme informatique 16

P

*Panel*

Un conteneur graphique utilisé dans les interfaces utilisateur pour regrouper et organiser d'autres composants graphiques, tels que des boutons, des zones de texte ou des images 21

*paramètres*

En prgroammation. Variables ou valeurs spécifiées lors de la création d'un objet d'une classe, utilisées pour initialiser les propriétés ou le comportement de cet objet 21

R

*requêtes SQL*

Instruction pour interagir avec une base de données 7

S

*script SQL*

Ensemble d'instructions pour effectuer des opérations sur une base de données, telles que la création de tables, l'insertion de données ou la modification de schémas 9

*SQL*

Abréviation de "Structured Query Language", langage de programmation utilisé pour communiquer avec et manipuler des bases de données relationnelles 11

T

*tuple*

Ensemble de plusieurs éléments regroupés ensemble 20

U

*URL*

Abréviation de "Uniform Resource Locator", une adresse utilisée pour localiser une ressource sur Internet, telle qu'une page web, un fichier ou un service 23

V

*varchar*

Abréviation de "variable character" en informatique, représentant un type de données qui permet de stocker des chaînes de caractères de longueur variable 10

W

*Wamp*

Acronyme de "Windows, Apache, MySQL, PHP", un ensemble de logiciels open-source utilisés ensemble pour créer un environnement de développement web sur des systèmes Windows 11

*Waterfall*

Approche où les étapes du projet sont suivies de manière séquentielle, sans retour en arrière 7

## Table des illustrations

[Figure 1 : Méthode Kanban 5](#_Toc168271400)

[Figure 2 : Commits GitHub du début du projet 9](#_Toc168271401)

[Figure 3 : Modèle Conceptuel des Données 10](#_Toc168271402)

[Figure 4 : Modèle Logique des Données (fait sur Looping) 10](#_Toc168271403)

[Figure 5 : Modèle Physique des Données (représentation depuis PHPMyAdmin) 11](#_Toc168271404)

[Figure 6 : Post-It 12](#_Toc168271405)

[Figure 7 : Logo de MyTaskManager 12](#_Toc168271406)

[Figure 8 : Maquette de l’interface de connexion 13](#_Toc168271407)

[Figure 9 : Maquette de l'interface de création d'un compte 13](#_Toc168271408)

[Figure 10 : Maquette de l'interface principale (version 1) 14](#_Toc168271409)

[Figure 11 : Maquette de l'interface principale (version finale) 14](#_Toc168271410)

[Figure 12 : Maquette de l'interface d'ajout d'une tâche 15](#_Toc168271411)

[Figure 13 : Fonctionnement du MVC schématisé 15](#_Toc168271412)

[Figure 14 : MVC de MyTaskManager 16](#_Toc168271413)

[Figure 15 : Méthode de hachage BCrypt 18](#_Toc168271414)

[Figure 16 : Evénement au clic du bouton « Créer un compte » 19](#_Toc168271415)

[Figure 17 : Méthode « Créer un compte » dans le Contrôleur 19](#_Toc168271416)

[Figure 18 : Méthode « Créer un compte » dans le Modèle 20](#_Toc168271417)

[Figure 19 : Méthode BCrypt.Verify() 21](#_Toc168271418)

[Figure 20 : Gestion des erreurs lors du login (dans la Vue) 21](#_Toc168271419)

[Figure 21 : Variante de l'apparence d'une tâche 22](#_Toc168271420)

[Figure 22 : Apparence d'une tâche dans l'application 22](#_Toc168271421)

[Figure 23 : Méthode MyTask\_MouseDown qui permet de rendre les tâches déplaçables 22](#_Toc168271422)

[Figure 24 : Méthode permettant d'ouvrir l'URL contenue dans la tâche 23](#_Toc168271423)

[Figure 25 : Glisser-Déposer de l'image de la tâche 23](#_Toc168271424)

[Figure 26 : Méthode pour convertir une image en tableau d'octets 24](#_Toc168271425)

[Figure 27 : Evénement au clic du bouton "Ajout d'une tâche" 24](#_Toc168271426)

## Webographie

***Youtube***

*How to Make Responsive Windows Form Application using Anchor Property*

<https://www.youtube.com/watch?v=Yvo7j39wgFg>

*Two ways of saving pictures or BLOB in the SQL Server Database using C#*

<https://www.youtube.com/watch?v=OPFv7-oFA1Y>

*How To Resize Controls When Form Changes Size In C# | Windows Forms*

<https://www.youtube.com/watch?v=TTsyUclt-XU>

***Wikipedia***

*Modèle-Vue-Contrôleur*

<https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur>

*Merise (informatique)*

<https://fr.wikipedia.org/wiki/Merise_(informatique)>

**Microsoft**

*How to: Manipulate Columns in the Windows Forms DataGridView Control*

<https://learn.microsoft.com/en-us/dotnet/desktop/winforms/controls/how-to-manipulate-columns-in-the-windows-forms-datagridview-control?view=netframeworkdesktop-4.8>

*SqlParameterCollection.AddWithValue(String, Object) Méthode*

<https://learn.microsoft.com/fr-fr/dotnet/api/system.data.sqlclient.sqlparametercollection.addwithvalue?view=netframework-4.8.1&viewFallbackFrom=dotnet-plat-ext-5.0>

*How to: Add to or Remove from a Collection of Controls at Run Time*

<https://learn.microsoft.com/en-us/dotnet/desktop/winforms/controls/how-to-add-to-or-remove-from-a-collection-of-controls-at-run-time?view=netframeworkdesktop-4.8>

*Copier une image d’une base de données directement vers un contrôle PictureBox C#*

<https://learn.microsoft.com/fr-fr/troubleshoot/developer/visualstudio/csharp/language-compilers/copy-image-database-picturebox>

**Stack Overflow**

*Removing dynamic controls from panel*

<https://stackoverflow.com/questions/13888558/removing-dynamic-controls-from-panel>

**Blogs**

*Introduction to List in C#*

<https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/c-sharp-list>

*La méthode des 6 étapes*

<https://www.cifc-vd.ch/docs/18_mois/ci_1/2._la_methode_des_6_etapes.pdf>

*Hashing and Salting Passwords in C#*

<https://medium.com/@imAkash25/hashing-and-salting-passwords-in-c-0ee223f07e20>

*How to Secure Passwords with BCrypt.NET*

<https://code-maze.com/dotnet-secure-passwords-bcrypt/>

*How to Hash Passwords with BCrypt in C#*

<https://www.claudiobernasconi.ch/2023/06/23/how-to-hash-passwords-with-bcrypt-in-csharp/>

*Les 6 phases de gestion de projet par la pratique : Etude de cas*

<https://blog-gestion-de-projet.com/phases-de-gestion-de-projet/>

*MVC Framework Introduction*

<https://www.geeksforgeeks.org/mvc-framework-introduction/>

*Architecture Modèle/Vue/Contrôleur*

<https://www.irif.fr/~carton/Enseignement/InterfacesGraphiques/Cours/Swing/mvc.html>

*C# Passage de variable entre forms*

<https://www.developpez.net/forums/d1988834/dotnet/developpement-windows/windows-forms/csharp-passage-variable-entre-forms/>

*MLD*

<https://www.base-de-donnees.com/mld/>

*Merise (informatique) - Définition*

<https://www.techno-science.net/definition/739.html>

*Initiation à la conception de bases de données relationnelles avec MERISE*

<https://ineumann.developpez.com/tutoriels/merise/initiation-merise/>

# Annexes

## Planification initiale

Une image contenant texte, capture d’écran, ligne, Police

Description générée automatiquement

Une image contenant texte, ligne, nombre, Police

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, ligne, logiciel, nombre

Description générée automatiquement

Une image contenant texte, ligne, nombre, Police

Description générée automatiquement

Une image contenant texte, ligne, nombre, logiciel

Description générée automatiquement

Une image contenant texte, ligne, nombre, logiciel

Description générée automatiquement

Une image contenant texte, ligne, Police, nombre

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, ligne, logiciel, nombre

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, ligne, nombre, Police

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, ligne, logiciel, nombre

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, ligne, nombre, logiciel

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, ligne, capture d’écran

Description générée automatiquement

Une image contenant texte, ligne, nombre, logiciel

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, ligne, nombre, Police

Description générée automatiquement

Une image contenant texte, ligne, nombre, Police

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

## Planification détaillée

## Cahier Des Charges

Une image contenant texte, capture d’écran, document, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, document, Police

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

1. Explication de l’architecture MVC :

   https://www.irif.fr/~carton/Enseignement/InterfacesGraphiques/Cours/Swing/mvc.html [↑](#footnote-ref-1)
2. Définition des User Stories : https://asana.com/fr/resources/user-stories [↑](#footnote-ref-2)
3. 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-3)
4. Pourquoi BCrypt est une solution pour le hachage des mots de passe :

   https://nordvpn.com/fr/blog/what-is-bcrypt/ [↑](#footnote-ref-4)