

## Compte rendu de l'atelier de professionnalisation 2

### Table des matières

<b>Rappel du contexte .....</b>	<b>2</b>
<b>Rappel de la mission globale à effectuer .....</b>	<b>2</b>
<b>Les étapes du projet .....</b>	<b>3</b>
Étape 1 : préparer l'environnement de travail et créer la base de données .....	3
1. Préparation de l'environnement de travail .....	3
2. Création de la base de données .....	3
3. Bilan sur les objectifs atteints .....	3
Étape 2 : dessiner les interfaces, structurer l'application en MVC, créer un dépôt, coder le visuel .....	4
1. Dessin des interfaces.....	4
2. Structure de l'application.....	5
3. Création du dépôt GitHub.....	6
4. Création du visuel de l'application .....	7
5. Bilan sur les objectifs atteints .....	8
Étape 3 : coder le modèle et les outils de connexion, générer la documentation technique .....	9
1. Codage du modèle de l'application.....	9
2. Codage des outils de connexion à la base de données .....	9
3. Génération de la documentation technique.....	9
4. Bilan sur les objectifs atteints .....	10
Étape 4 : coder les fonctionnalités de l'application à partir des cas d'utilisation .....	11
1. Codage des fonctionnalités de l'application.....	11
2. Finitions de l'application .....	15
3. Mise à jour de la documentation technique.....	15
4. Bilan sur les objectifs atteints .....	16
Étape 5 : créer une documentation utilisateur en vidéo .....	16
1. Tournage de la vidéo de documentation utilisateur.....	16
2. Montage de la vidéo de documentation utilisateur.....	17
3. Bilan sur les objectifs atteints .....	18
Étape 6 : gérer le déploiement, rédiger le compte rendu, créer la page du portfolio dédiée à la mission.....	19
1. Création d'un installateur pour l'application .....	19
2. Génération du script complet de la base de données et de son utilisateur .....	19
3. Rédaction du compte rendu du projet .....	20
4. Création d'une page dédiée au projet dans le portfolio.....	20
5. Bilan sur les objectifs atteints .....	20
<b>Bilan final.....</b>	<b>21</b>

## Rappel du contexte

InfoTech Services 86 (ITS 86) est une Entreprise de Services Numériques (ESN) fondée en 2002, spécialisée dans le développement informatique (web, mobile, logiciels), l'infogérance, l'hébergement de sites web, et l'ingénierie système et réseau. L'entreprise s'adresse principalement aux TPE et PME, avec une équipe de 32 collaborateurs partageant des valeurs fortes comme l'honnêteté, la transparence et la convivialité.

ITS 86 s'organise autour de deux pôles :

- **Le Pôle Développement** : création d'applications sur mesure (C#, Java, PHP, Symfony, etc.), gestion de bases de données, solutions d'hébergement sur serveurs dédiés.
- **Le Pôle Systèmes et Réseaux** : infogérance, support technique, virtualisation, cybersécurité, installation de serveurs Linux/Windows, administration réseau.

L'entreprise accompagne ses clients à chaque étape de leurs projets informatiques : conseil, conception, développement, intégration, et maintenance. Elle s'inscrit dans une démarche qualité basée sur les normes ISO 9001, ISO 27001, ITIL et Prince2, avec des outils assurant la traçabilité et la fiabilité des prestations fournies.

D'autre part, MediaTek86 est un réseau qui gère les médiathèques de la Vienne, leur affectation à un service et leurs absences, et pour lequel ITS 86 doit réaliser des prestations dans le domaine du développement d'application.

## Rappel de la mission globale à effectuer

Lors de ce projet je dois travailler en tant que technicien développeur junior pour ITS 86 sur le développement d'une application de bureau en C# qui exploite une base de données MySQL et a pour but final de permettre la gestion du personnel de chaque médiathèque du réseau MédiaTek86, ainsi que celle des absences de ce même personnel. La création de la base de données MySQL, tout comme la gestion de la sauvegarde régulière des avancées de l'application, me sont d'ailleurs également confiées.

# Les étapes du projet

## Étape 1 : préparer l'environnement de travail et créer la base de données

### 1. Préparation de l'environnement de travail

Afin de mener à bien ce projet de développement d'application, j'ai tout d'abord vérifié que je disposais bien des logiciels Visual Studio Enterprise 2022 en tant qu'IDE pour le code, WampServer en tant que serveur pour héberger la base de données à créer, ainsi que Looping en tant que logiciel de modélisation permettant de visualiser le schéma conceptuel de données fourni.

Tous ces logiciels étaient bien installés et correctement configurés sur mon poste de travail.

### 2. Création de la base de données

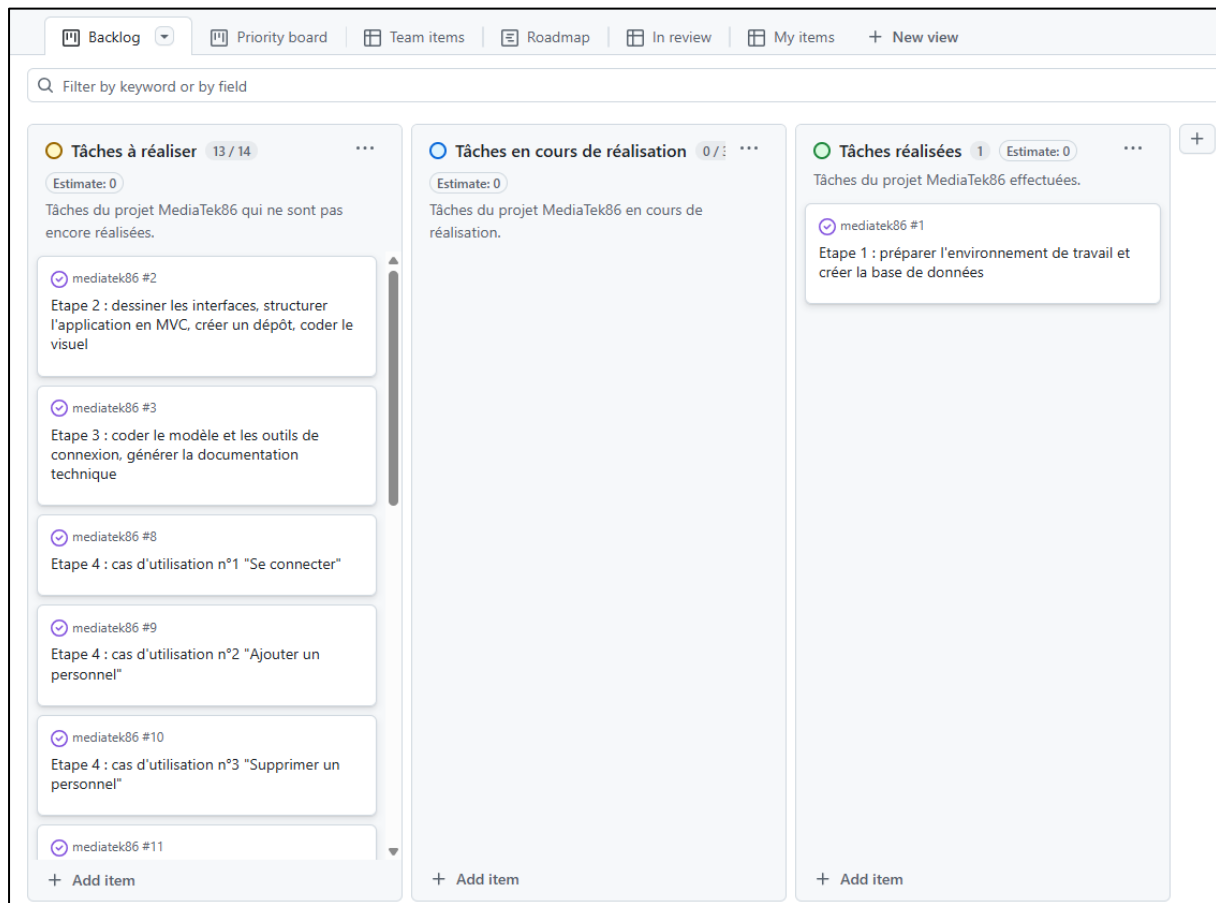
Après avoir récupéré le script SQL correspondant schéma conceptuel de données fourni, j'ai pu créer la base de données « mediatek86 » sous MySQL, y ajouter les tables souhaitées (responsable, motif, service, personnel et absence) et leurs champs respectifs, ainsi que créer l'utilisateur « mediatek86user » ayant les droits d'accès à cette base de données.

La base de données pour l'application était alors prête.

### 3. Bilan sur les objectifs atteints

Une fois cette première étape finie l'environnement de travail était complètement prêt, aussi bien au niveau de l'IDE qu'au niveau de la base de données. Le développement de l'application pouvait alors réellement commencer.

Avancement du « project » (kanban) à ce stade du projet :



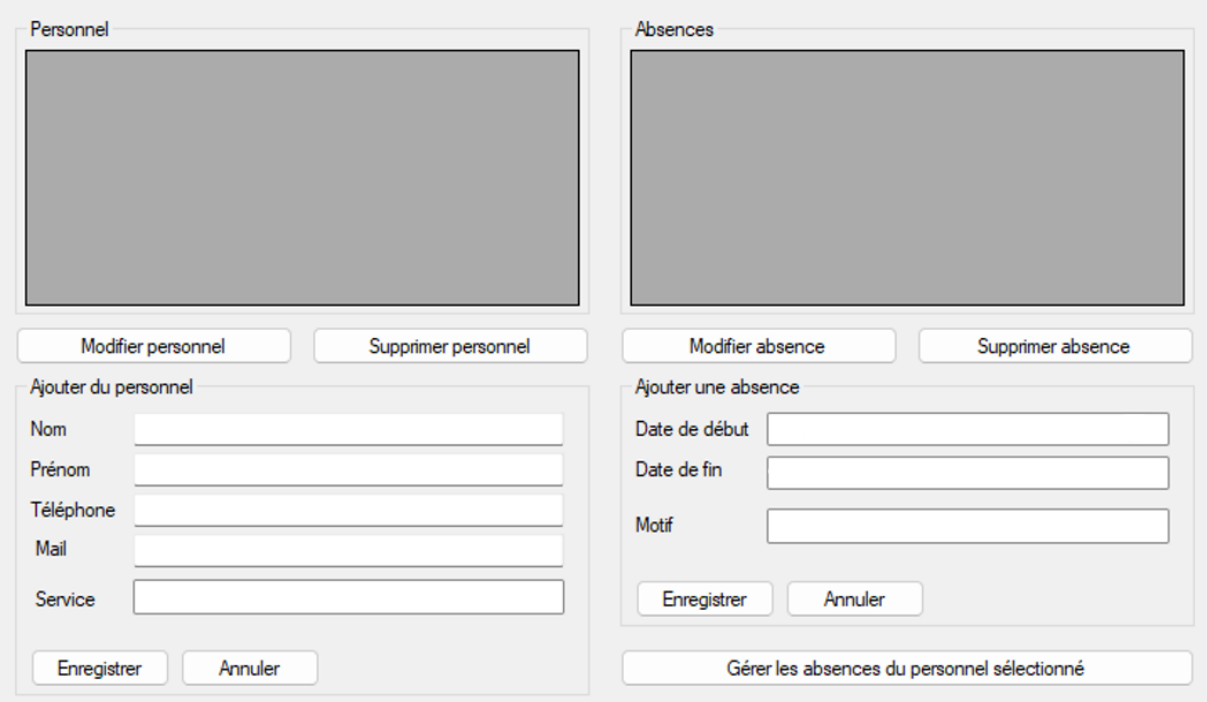
Étape 2 : dessiner les interfaces, structurer l'application en MVC, créer un dépôt, coder le visuel

## 1. Dessin des interfaces

Après étude du dossier documentaire fourni, j'ai réalisé une maquette pour chacune des interfaces graphiques de l'application à l'aide d'un outil de maquettage.

Maquette de l'interface de connexion de l'application :

Maquette de l'interface principale de l'application :



Maquette de l'interface principale de l'application, divisée en deux sections principales : Personnel et Absences.

**Personnel**

- Zone de liste (grisée).
- Boutons : Modifier personnel, Supprimer personnel.
- Section Ajouter du personnel :

  - Formulaires : Nom, Prénom, Téléphone, Mail, Service.
  - Boutons : Enregistrer, Annuler.

**Absences**

- Zone de liste (grisée).
- Boutons : Modifier absence, Supprimer absence.
- Section Ajouter une absence :

  - Formulaires : Date de début, Date de fin, Motif.
  - Boutons : Enregistrer, Annuler.

- Bouton : Gérer les absences du personnel sélectionné.

## 2. Structure de l'application

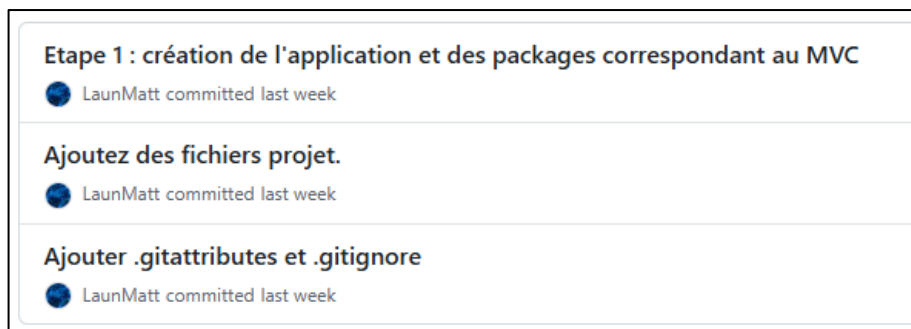
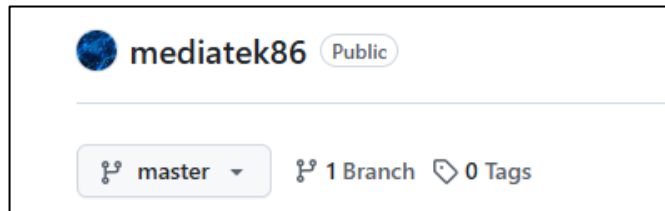
La maquette de l'application étant réalisée, j'ai ensuite pu créer un nouveau projet sous Visual Studio Enterprise 2022 pour l'application « MediaTek86 ».

Une fois cela fait, j'ai créé les packages correspondant au MVC (model, view et controller)

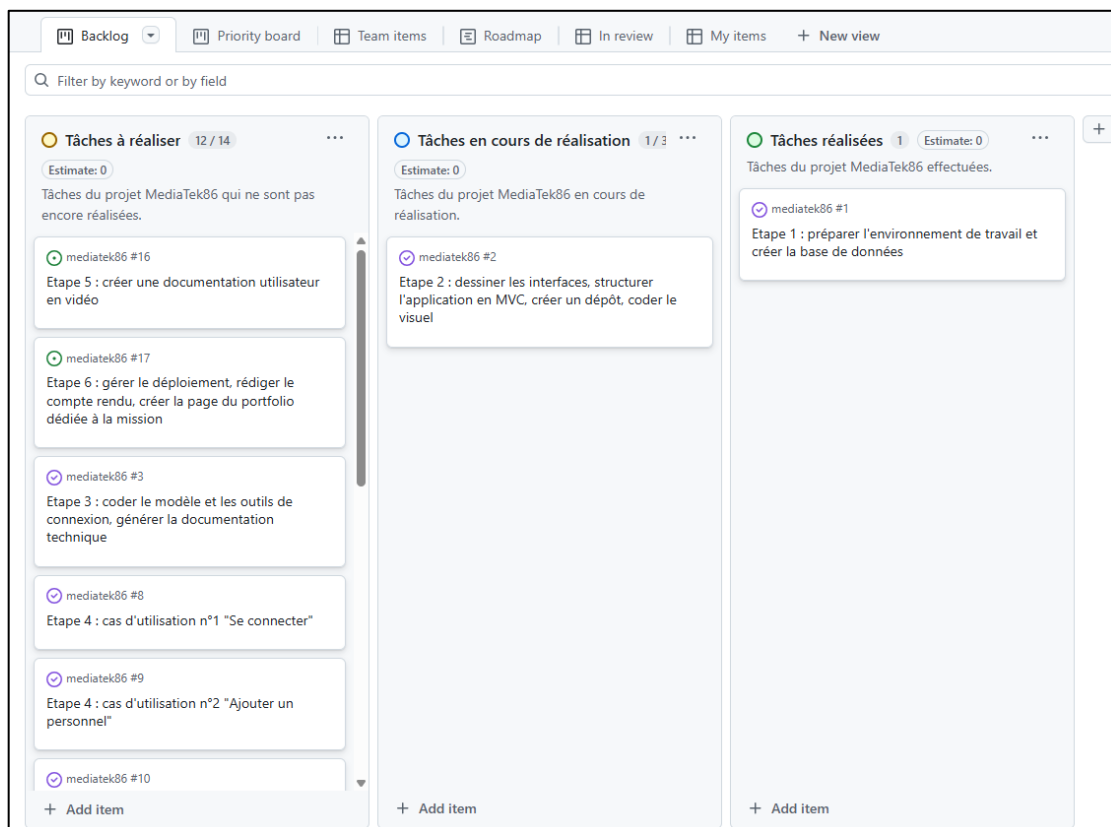


### 3. Création du dépôt GitHub

Les packages de l'application étant alors créés, j'ai pu créer un dépôt distant pour l'application sur GitHub et faire une première sauvegarde du projet.



J'ai également créé un « project » (kanban) et y ai inséré toutes les issues correspondant aux tâches à réaliser, afin d'avoir un suivi du projet étape par étape.



## 4. Création du visuel de l'application

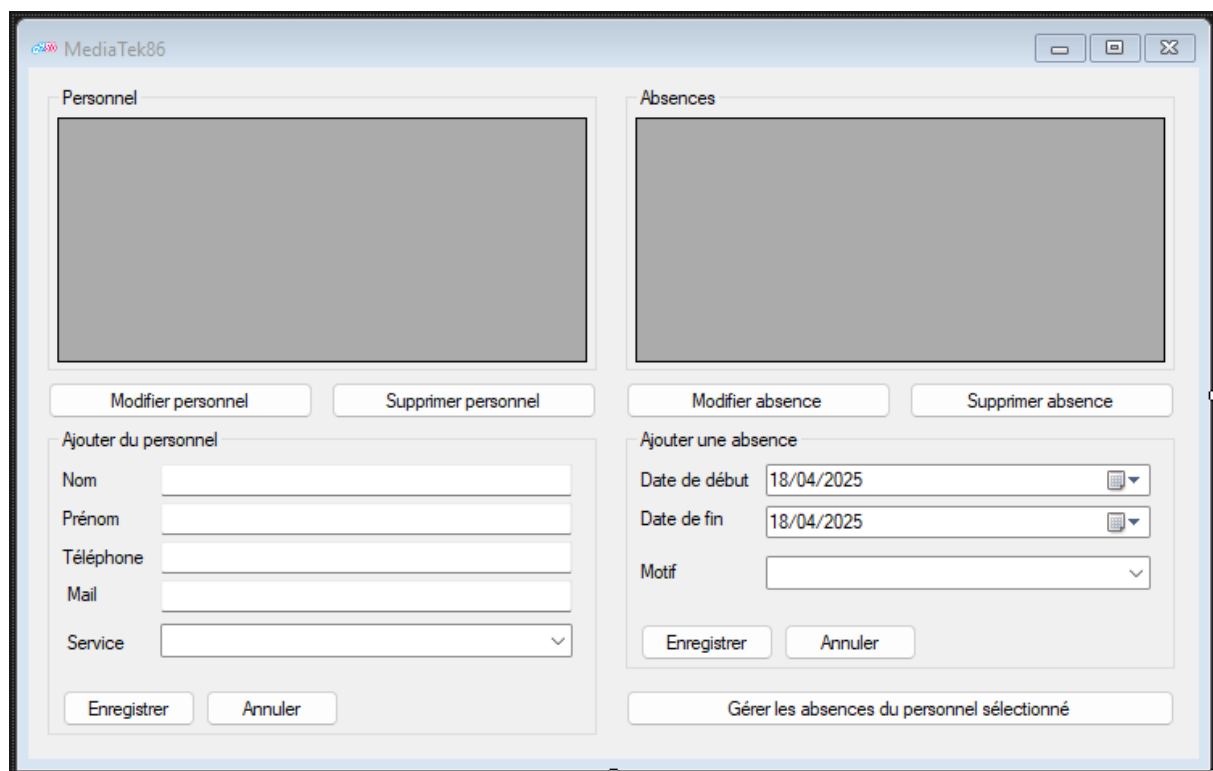
Une fois le « project » (kanban) créé, j'ai pu coder la partie Vue de l'application (uniquement le visuel des interfaces).

L'interface de connexion de l'application :



The screenshot shows a small window titled "Connection" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there are two text input fields: the first is labeled "Login" and the second is labeled "Password". Below these fields is a button labeled "Se connecter".

L'interface principale de l'application :

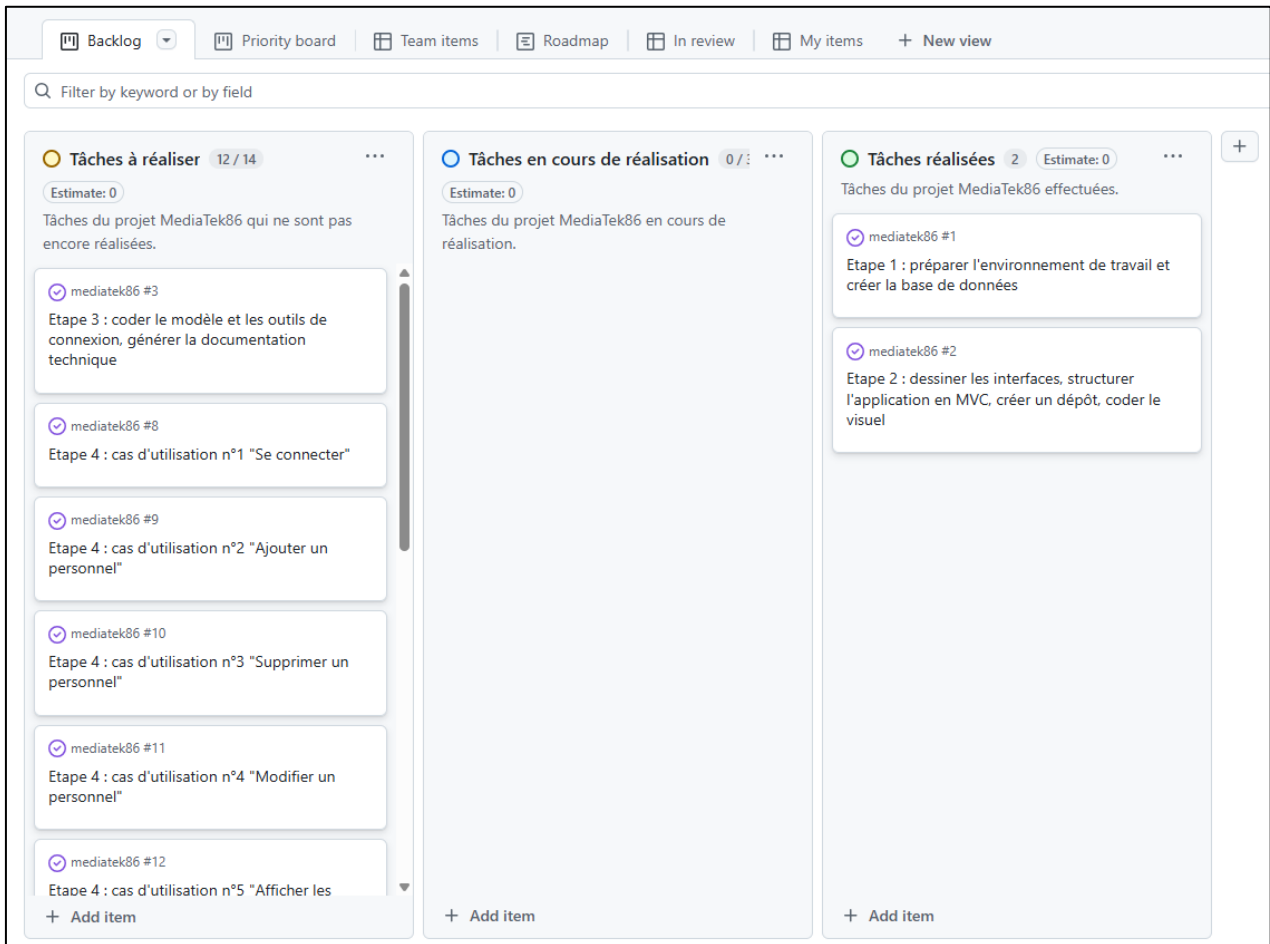


The screenshot shows the main application window titled "MediaTek86". It features a split layout with two main panels. The left panel is titled "Personnel" and contains a large empty rectangular area for a list. Below this area are two buttons: "Modifier personnel" and "Supprimer personnel". At the bottom of the left panel is a section titled "Ajouter du personnel" with input fields for "Nom", "Prénom", "Téléphone", "Mail", and a dropdown menu for "Service". There are "Enregistrer" and "Annuler" buttons at the bottom of this section. The right panel is titled "Absences" and also contains a large empty rectangular area. Below it are buttons for "Modifier absence" and "Supprimer absence". At the bottom of the right panel is a section titled "Ajouter une absence" with input fields for "Date de début" (set to 18/04/2025), "Date de fin" (set to 18/04/2025), and a dropdown for "Motif". There are "Enregistrer" and "Annuler" buttons here as well. A wide button at the very bottom of the right panel is labeled "Gérer les absences du personnel sélectionné".

## 5. Bilan sur les objectifs atteints

Une fois cette seconde étape finie, les interfaces graphiques de l'application désormais structurée en MVC étaient réalisées, tout comme le dépôt distant de l'application sur GitHub et les premières sauvegardes de celle-ci ont été créées. La création des premières classes de l'application pouvait alors débuter.

Avancement du « project » (kanban) à ce stade du projet :

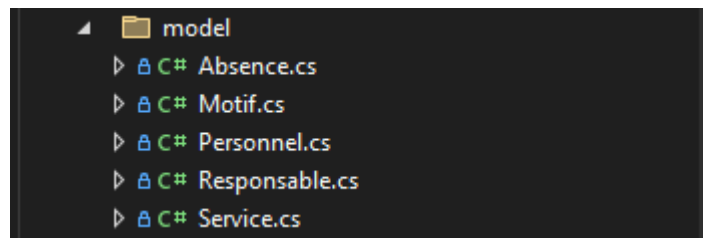




## Étape 3 : coder le modèle et les outils de connexion, générer la documentation technique

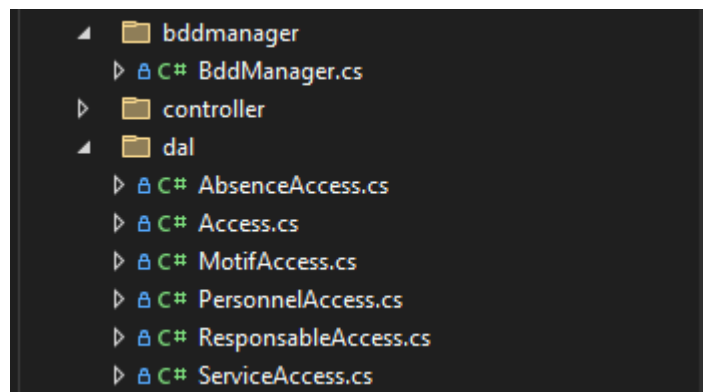
### 1. Codage du modèle de l'application

Lors de cette étape 3, j'ai tout d'abord créé les classes demandées au niveau du package model et correspondant aux tables de la base de données « mediatek86 ».



### 2. Codage des outils de connexion à la base de données

J'ai ensuite créé les packages bddmanager et dal demandés, puis j'ai également créé les classes demandées au niveau de ces deux packages.



### 3. Génération de la documentation technique

Une fois les packages et classes demandés précédemment créés, j'ai pu rédiger les commentaires normalisés des premières méthodes et classes composant l'application, puis j'ai généré une première version de la documentation technique de l'application à l'aide de l'outil Sandcastle, avant de la sauvegarder avec le reste du projet sur le dépôt GitHub.

## 4. Bilan sur les objectifs atteints

Une fois cette troisième étape finie, les classes du modèle ainsi que les outils de connexion à la base de données « mediatek86 » étaient codés et une première version de la documentation technique a été générée. Les fonctionnalités de l'application pouvaient alors commencer à être codées.

Avancement du « project » (kanban) à ce stade du projet :

The screenshot shows a Kanban board interface with three columns. The top navigation bar includes tabs for 'Backlog', 'Priority board', 'Team items', 'Roadmap', 'In review', 'My items', and a '+ New view' button. A search bar below the tabs says 'Filter by keyword or by field'.


- Tâches à réaliser (11 / 14)**: This column contains six task cards, all with a status of 'Etape 4'. The tasks are:
  - mediatek86 #8: Etape 4 : cas d'utilisation n°1 "Se connecter"
  - mediatek86 #9: Etape 4 : cas d'utilisation n°2 "Ajouter un personnel"
  - mediatek86 #10: Etape 4 : cas d'utilisation n°3 "Supprimer un personnel"
  - mediatek86 #11: Etape 4 : cas d'utilisation n°4 "Modifier un personnel"
  - mediatek86 #12: Etape 4 : cas d'utilisation n°5 "Afficher les absences"
  - mediatek86 #13: Etape 4 : cas d'utilisation n°6 "Ajouter une absence"
- Tâches en cours de réalisation (0 / 1)**: This column is currently empty.
- Tâches réalisées (3)**: This column contains three task cards, all with a status of 'Etape 1', 'Etape 2', or 'Etape 3'. The tasks are:
  - mediatek86 #1: Etape 1 : préparer l'environnement de travail et créer la base de données
  - mediatek86 #2: Etape 2 : dessiner les interfaces, structurer l'application en MVC, créer un dépôt, coder le visuel
  - mediatek86 #3: Etape 3 : coder le modèle et les outils de connexion, générer la documentation technique

Each column has an 'Add item' button at the bottom.

## Étape 4 : coder les fonctionnalités de l'application à partir des cas d'utilisation

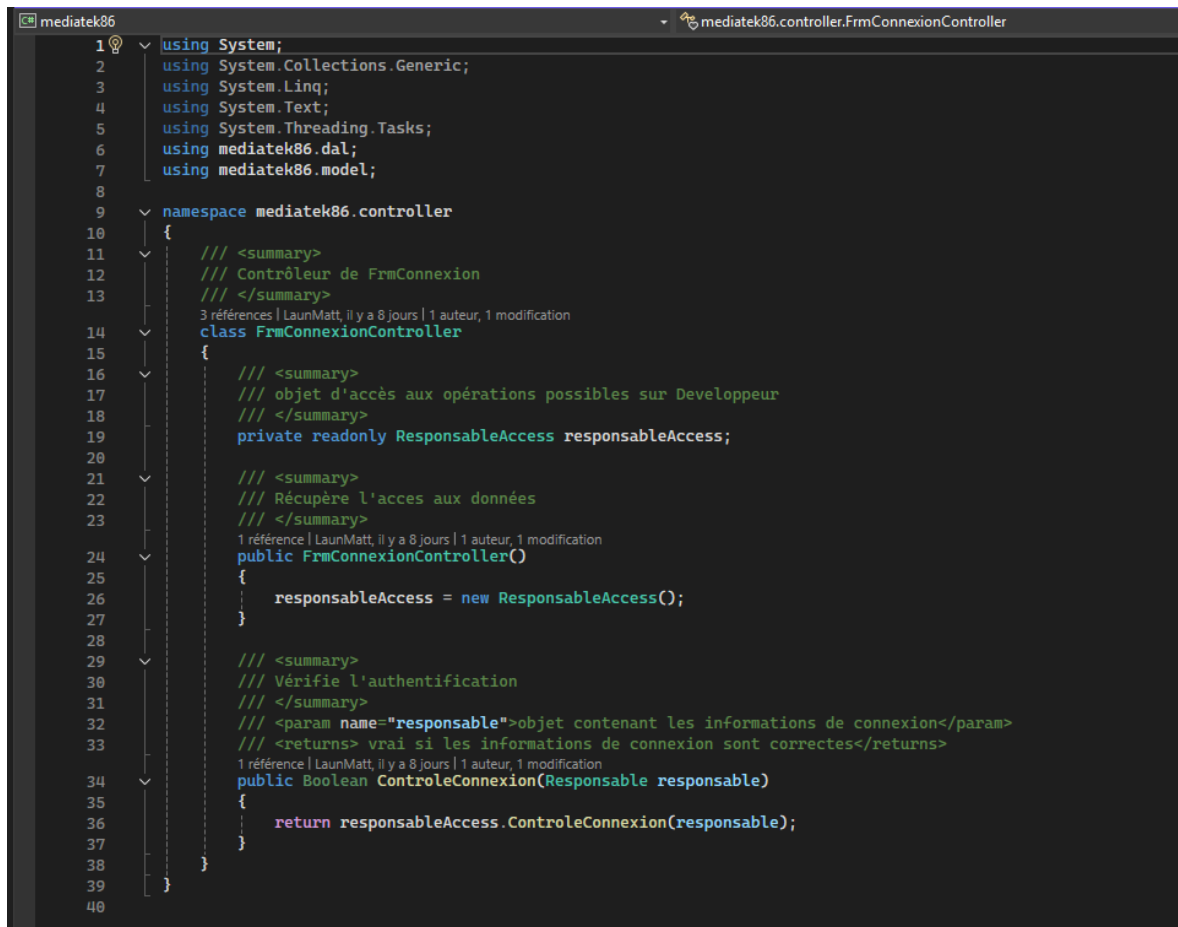
### 1. Codage des fonctionnalités de l'application

Lors de cette étape 4 et afin que l'utilisateur « responsable » puisse se connecter à l'application, j'ai tout d'abord écrit le code de la classe « FrmConnexion » liée à l'interface de connexion de l'application :



```
1 using mediatek86.controller;
2 using System.Windows.Forms;
3 using mediatek86.model;
4 using System;
5
6 namespace mediatek86.view
7 {
8     /// <summary>
9     /// Première fenêtre de l'application permettant à l'utilisateur responsable de se connecter à l'application.
10    /// </summary>
11    public partial class FrmConnexion : Form
12    {
13        /// <summary>
14        /// Contrôleur de la fenêtre
15        /// </summary>
16        private FrmConnexionController controller;
17
18        /// <summary>
19        /// Méthode initialisant les composants de FrmConnexion.
20        /// </summary>
21        public FrmConnexion()
22        {
23            InitializeComponent();
24            Init();
25        }
26
27        /// <summary>
28        /// Initialisations :
29        /// Création du contrôleur
30        /// </summary>
31        private void Init()
32        {
33            controller = new FrmConnexionController();
34        }
35
36        /// <summary>
37        /// Demande au contrôleur de contrôler l'authentification
38        /// </summary>
39        /// <param name="sender">sender</param>
40        /// <param name="e">e</param>
41        private void btnConnection_Click(object sender, System.EventArgs e)
42        {
43            String login = txtLogin.Text;
44            String pwd = txtPassword.Text;
45            if (String.IsNullOrEmpty(login) || String.IsNullOrEmpty(pwd))
46            {
47                MessageBox.Show("Tous les champs doivent être remplis.", "Information");
48            }
49            else
50            {
51                Responsable responsable = new Responsable(login, pwd);
52                if (controller.ControleConnexion(responsable))
53                {
54                    FrmMediatek86 frm = new FrmMediatek86();
55                    frm.ShowDialog();
56                }
57                else
58                {
59                    MessageBox.Show("Authentification incorrecte", "Alerte");
60                }
61            }
62        }
63    }
64 }
65
```

J'ai pu ensuite créer la classe « FrmConnexionController » dans le package controller, chargée de faire le lien entre « FrmConnexion » et le modèle, ainsi que le dal :



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using mediatek86.dal;
7  using mediatek86.model;
8
9  namespace mediatek86.controller
10 {
11     /// <summary>
12     /// Contrôleur de FrmConnexion
13     /// </summary>
14     3 références | LaunMatt, il y a 8 jours | 1 auteur, 1 modification
15     class FrmConnexionController
16     {
17         /// <summary>
18         /// objet d'accès aux opérations possibles sur Developpeur
19         /// </summary>
20         private readonly ResponsableAccess responsableAccess;
21
22         /// <summary>
23         /// Récupère l'accès aux données
24         /// </summary>
25         1 référence | LaunMatt, il y a 8 jours | 1 auteur, 1 modification
26         public FrmConnexionController()
27         {
28             responsableAccess = new ResponsableAccess();
29         }
30
31         /// <summary>
32         /// Vérifie l'authentification
33         /// </summary>
34         /// <param name="responsable">objet contenant les informations de connexion</param>
35         /// <returns>vrai si les informations de connexion sont correctes</returns>
36         1 référence | LaunMatt, il y a 8 jours | 1 auteur, 1 modification
37         public Boolean ControleConnexion(Responsable responsable)
38         {
39             return responsableAccess.ControleConnexion(responsable);
40         }
41     }
42 }
```

Puis, j'ai logiquement créé la classe « ResponsableAccess » afin de créer une liaison avec les données du responsable de la base de données et ai écrit le code de cette même classe.

Il ne restait plus qu'à ajouter la ligne de code « Application.Run(new view.FrmConnexion()); » dans la classe « Program », afin que la fenêtre de connexion soit lancée en premier au démarrage de l'application.

Une fois cela fait, j'ai également dû coder la méthode d'initialisation de la classe « FrmMediatek86 », à savoir la méthode « Init » :

```
/// <summary>
/// Initialisations :
/// Création du contrôleur et remplissage des listes
/// </summary>
1 référence | LaunMatt, il y a 3 jours | 1 auteur, 3 modifications
private void Init()
{
    controller = new FrmMediatek86Controller();
    RemplirListePersonnel();
    RemplirListeServices();
    RemplirListeMotifs();
    EnCoursDeModifPersonnel(false);
    gbxAjoutModifAbsence.Enabled = false;
    btnModifAbsence.Enabled = false;
    btnSupprimerAbsence.Enabled = false;
}
```

Ensuite, afin de permettre l'ajout d'un utilisateur une fois que le responsable s'est connecté à l'application, j'ai notamment écrit les méthodes des classes « FrmMediatek86Controller », « PersonnelAccess », « ServiceAccess » et « FrmMediatek86 » permettant de récupérer et d'afficher la liste du personnel et de leur service (« GetLePersonnel », « GetLesServices », « RemplirListePersonnel » et « RemplirListeServices ») et d'autre part, la méthode permettant l'enregistrement de l'ajout d'un personnel (« btnEnregistrerPersonnel\_Click »), ainsi que les méthodes liées dans les packages controller et dal (« AddPersonnel »)..

Après cela, de façon à rendre possible la suppression d'un personnel, j'ai codé la méthode de la classe « FrmMediatek86 » permettant cette suppression (« btnSupprimerPersonnel\_Click »), ainsi que les méthodes liées dans les packages controller et dal (« DelPersonnel »).

Pour ce qui est de la possibilité de modifier un personnel, j'ai dû écrire la méthode permettant cette action dans la classe « FrmMediatek86 » (« btnModifPersonnel\_Click »), les méthodes liées dans les packages controller et dal (« UpdatePersonnel ») et celle permettant à l'application de savoir si l'on est en cours de modification d'un personnel ou non (« EnCoursDeModifPersonnel »).

Par ailleurs, j'ai aussi codé la courte méthode permettant l'annulation d'un ajout ou d'une modification d'un personnel dans la classe « FrmMediatek86 » :

```
/// <summary>
/// Annule la demande d'ajout ou de modification du personnel
/// Vide les zones de saisie du personnel
/// </summary>
/// <param name="sender">sender</param>
/// <param name="e">e</param>
1 référence | LaunMatt, il y a 3 jours | 1 auteur, 2 modifications
private void btnAnnulerPersonnel_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Voulez-vous vraiment annuler ?", "Confirmation", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        EnCoursDeModifPersonnel(false);
    }
}
```

Concernant maintenant les absences du personnel, et plus particulièrement l’affichage de ces mêmes absences, j’ai notamment codé les méthodes des classes « FrmMediatek86Controller », « AbsenceAccess », « MotifAccess » et « FrmMediatek86 » permettant de récupérer, puis d’afficher la liste des absences et de leur motif (« GetLesAbsences », « GetLesMotifs », « RemplirListeAbsences » et « RemplirListeMotifs »). J’ai par la même occasion écrit le code de la méthode « btnGestionAbsence\_Click » dans la classe « FrmMediatek86 », afin que le programme affiche la liste des absences du personnel et les motifs de ces dernières une fois que l’utilisateur responsable ait cliqué sur le bouton prévu à cet effet dans l’application.

Après cela, afin de permettre l’ajout d’absences j’ai écrit le code de la méthode permettant l’enregistrement de l’ajout d’une absence (« btnEnregistrerAbsence\_Click »), qui vérifie d’ailleurs si la date de début de l’absence entrée est bien antérieure à la date de fin de cette même absence, ainsi que les méthodes liées dans les packages controller et dal (« AddAbsence »). J’ai également dû coder la méthode « AbsenceChevauche » qui est importante dans le programme, car elle permet de vérifier si l’absence que l’on souhaite ajouter existe déjà ou non dans la liste des absences :

```

/// <summary>
/// Vérifie s'il y a une superposition d'absences pour le personnel sélectionné
/// </summary>
/// <param name="dateDebut"> date de début de l'absence</param>
/// <param name="dateFin">date de fin de l'absence</param>
/// <param name="idPersonnel">identifiant du personnel de l'absence</param>
/// <param name="ancienneDateDebut">ancienne date de début de l'absence</param>
/// <param name="ancienneDateFin">ancienne date de fin de l'absence</param>
/// <returns>vrai ou faux</returns>
2 références | LaunMatt, il y a 3 jours | 1 auteur, 2 modifications
private bool AbsenceChevauche(DateTime dateDebut, DateTime dateFin, int idPersonnel, DateTime? ancienneDateDebut = null, DateTime? ancienneDateFin = null)
{
    List<Absence> absencesExistantes = controller.GetLesAbsences(idPersonnel);

    foreach (Absence absence in absencesExistantes)
    {
        if (ancienneDateDebut != null && ancienneDateFin != null && absence.Datedebut == ancienneDateDebut && absence.Datefin == ancienneDateFin)
        {
            continue;
        }
        if (!(dateFin < absence.Datedebut || dateDebut > absence.Datefin))
        {
            return true;
        }
    }
    return false;
}

```

Puis, de manière à rendre possible la suppression d’une absence, j’ai codé la méthode de la classe « FrmMediatek86 » permettant cette suppression (« btnSupprimerAbsence\_Click »), ainsi que les méthodes liées dans les packages controller et dal (« DelAbsence »). J’ai de même codé la méthode « DelAbsencesPersonnel » dans la classe « FrmMediatek86Controller », ainsi que la méthode correspondante dans la classe « PersonnelAccess », afin que les absences d’un personnel soient aussi supprimées lorsque ce même personnel est supprimé par l’utilisateur responsable.

Enfin dans le but de permettre la modification d’une absence, j’ai écrit la méthode permettant cette action dans la classe « FrmMediatek86 » (« btnModifAbsence\_Click ») ainsi que les méthodes liées dans les packages controller et dal (« UpdateAbsence ») et celle permettant à l’application de savoir si l’on est en cours de modification d’une absence ou non (« EnCoursDeModifAbsence »). J’ai également mis à jour les méthodes

« btnEnregistrerAbsence\_Click » et « AbsenceChevauche » de la classe « FrmMediatek86 » afin que le programme vérifie que le créneau de l'absence après modification existe déjà ou non dans la liste des absences et que sa nouvelle date de début soit bien antérieure à sa nouvelle date de fin.

D'ailleurs, j'ai aussi écrit la courte méthode permettant l'annulation d'un ajout ou d'une modification d'une absence dans la classe « FrmMediatek86 » :

```
/// <summary>
/// Annule la demande d'ajout ou de modification de l'absence
/// Vide les zones de saisie de l'absence
/// </summary>
/// <param name="sender">sender</param>
/// <param name="e">e</param>
1 référence | LaunMatt, il y a 3 jours | 1 auteur, 2 modifications
private void btnAnnulerAbsence_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Voulez-vous vraiment annuler ?", "Confirmation", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        EnCoursDeModifAbsence(false);
    }
}
```

## 2. Finitions de l'application

Une fois les fonctionnalités de l'application entièrement codées, j'ai pu ajouter une icône pour les interfaces de l'application.

Ikône créée présente dans le coin supérieur gauche de l'interface de connexion et de l'interface principale de l'application MediaTek86 :



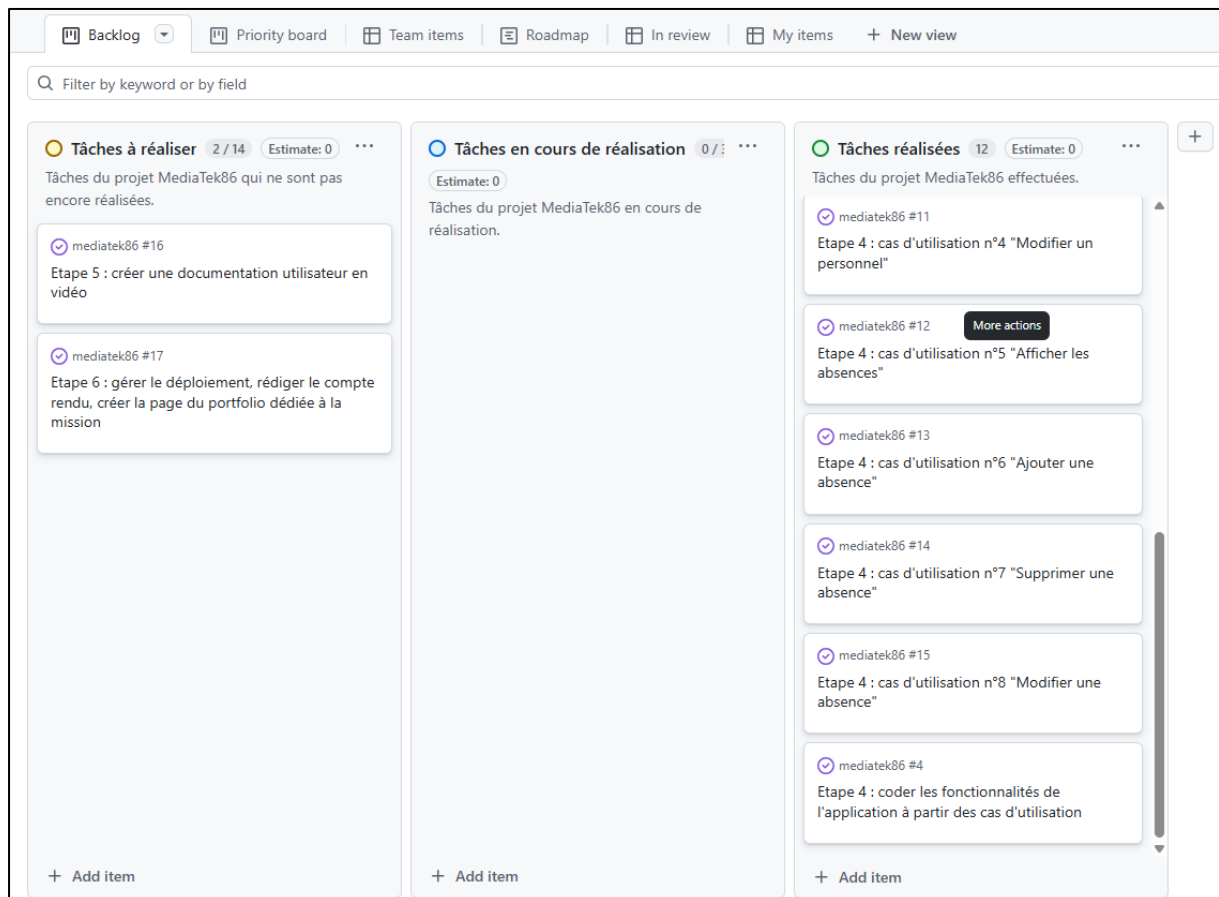
## 3. Mise à jour de la documentation technique

Les finitions de l'application étant terminées, j'ai pu vérifier et corriger ou ajouter l'ensemble des commentaires normalisés de l'application, dans le but ensuite de générer la nouvelle documentation technique mise à jour de l'application, comportant l'ensemble des informations sur les classes, méthodes et paramètres.

## 4. Bilan sur les objectifs atteints

Une fois cette quatrième étape finie, l'application était complètement fonctionnelle, les détails étaient quant à eux finis et la documentation technique de l'application était parfaitement à jour. Le tournage de la vidéo de documentation utilisateur pouvait alors débuter.

Avancement du « project » (kanban) à ce stade du projet :



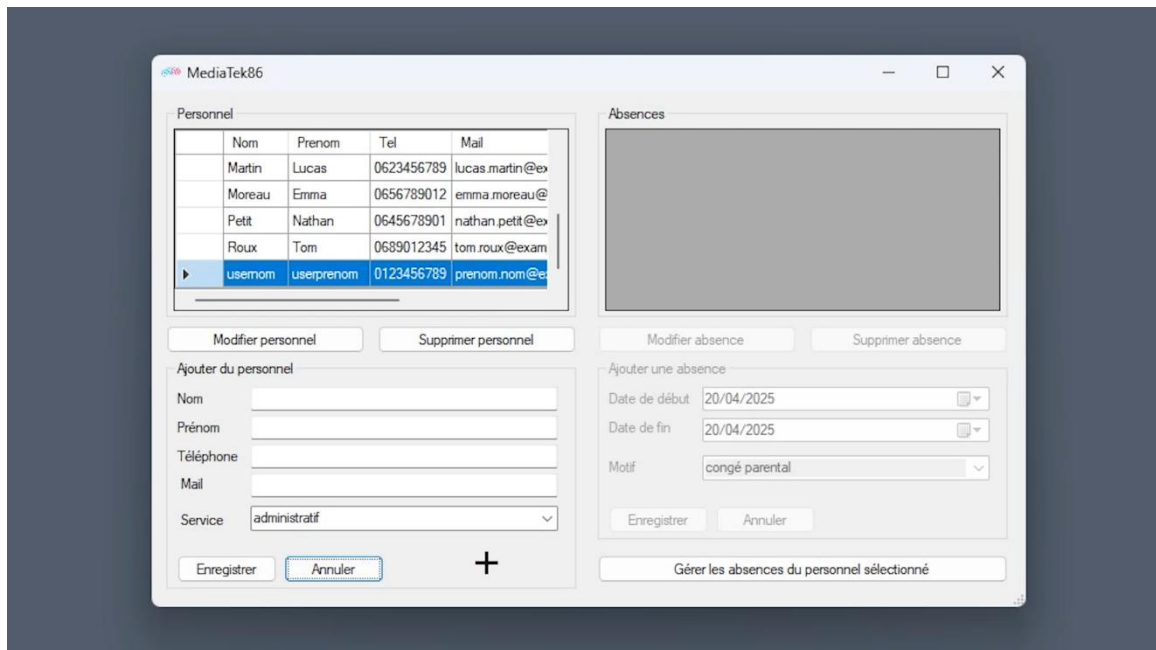
## Étape 5 : créer une documentation utilisateur en vidéo

### 1. Tournage de la vidéo de documentation utilisateur

Dans le but de réaliser une documentation utilisateur de l'application MediaTek86 au format vidéo, j'ai dû premièrement tourner la vidéo en question. Il a donc fallu que je prépare le matériel audio et le logiciel nécessaires à l'enregistrement.



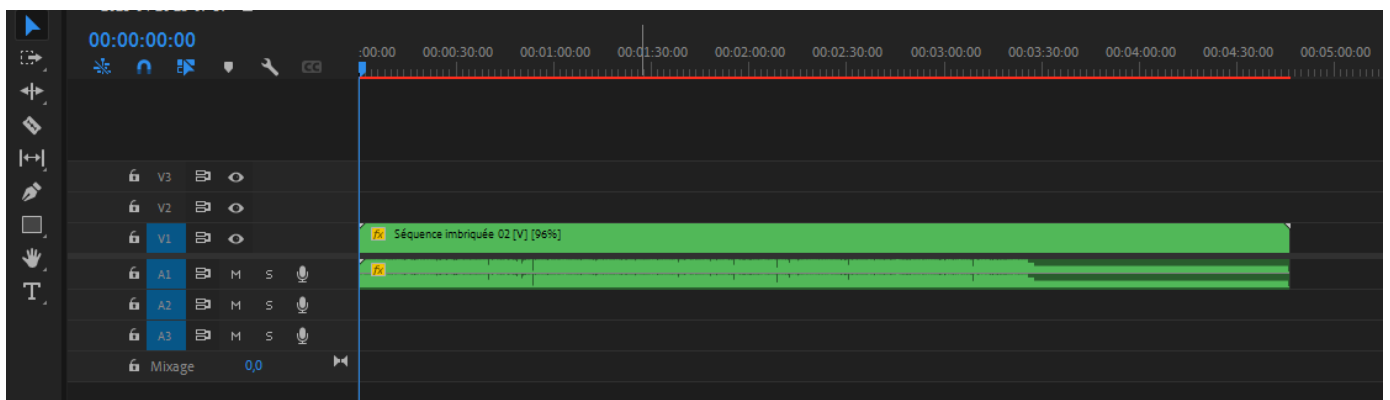
Vue de l'enregistrement de l'écran lors de la présentation de l'application :



## 2. Montage de la vidéo de documentation utilisateur

Après avoir tourné la vidéo, j'ai dû réaliser le montage de cette dernière pour synchroniser les différentes sources d'enregistrement.

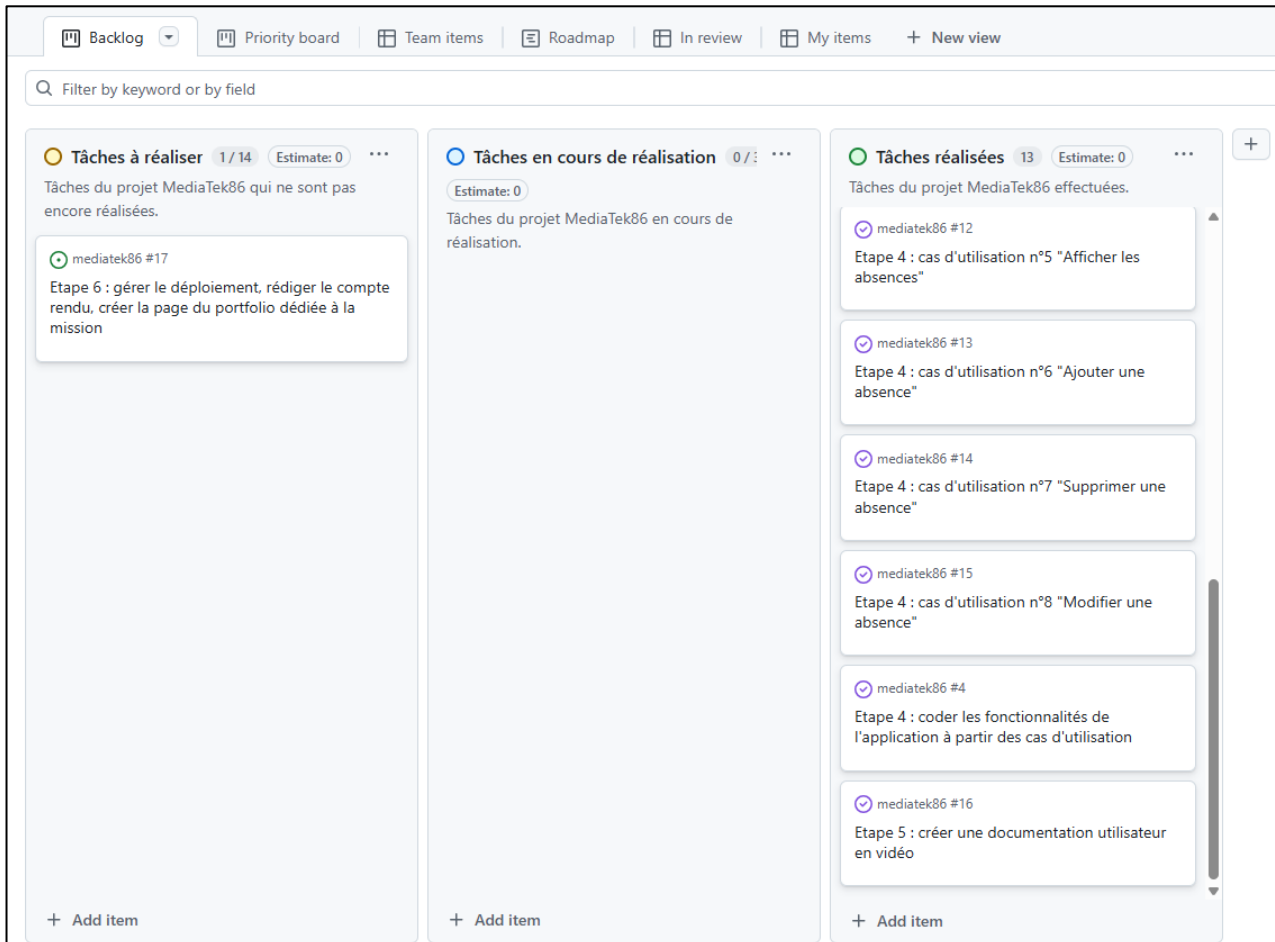
Vue du logiciel de montage, lors du montage de la documentation utilisateur au format vidéo :



### 3. Bilan sur les objectifs atteints

Une fois cette cinquième étape finie, la vidéo de documentation utilisateur présentant le fonctionnement et le contenu de l'application était entièrement prête. Il était alors possible de débiter la gestion du déploiement de l'application, la rédaction du compte rendu du projet, ainsi que la création de la page du portfolio dédiée à ce même projet.

Avancement du « project » (kanban) à ce stade du projet :



## Étape 6 : gérer le déploiement, rédiger le compte rendu, créer la page du portfolio dédiée à la mission

### 1. Création d'un installateur pour l'application

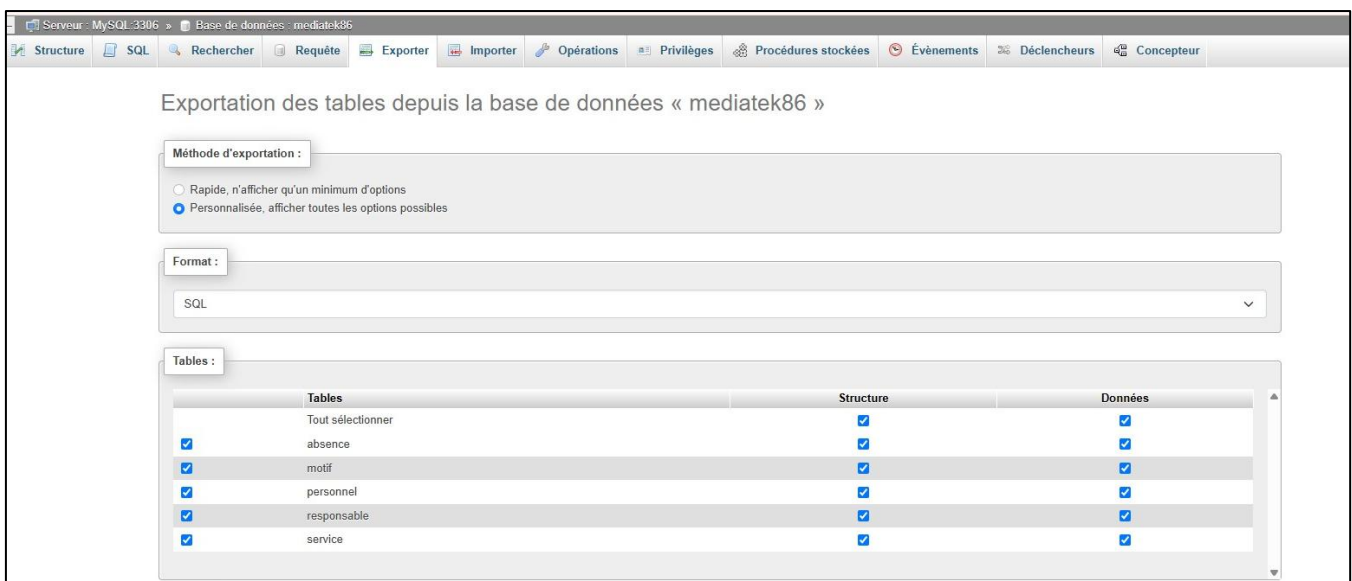
Lors de cette sixième et dernière étape, il a tout d'abord fallu que je crée un installateur pour l'application en suivant le mode opératoire fourni et dans le but de pouvoir démarrer l'application sans avoir besoin de Visual Studio.

Icône créée de l'application MediaTek86 une fois installée :



### 2. Génération du script complet de la base de données et de son utilisateur

Une fois l'installateur créé, j'ai ensuite généré le script complet de la base de données « mediatek86 » ainsi que de son contenu (insert), dans un document à l'aide de l'onglet d'exportation de PHPMyAdmin. J'ai aussi généré le script de création de l'utilisateur « mediatek86user » ayant les droits d'accès à la base de données, dans ce même document.



Tables	Structure	Données
<input checked="" type="checkbox"/> Tout sélectionner	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> absence	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> motif	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> personnel	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> responsable	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> service	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

### 3. Rédaction du compte rendu du projet

Après avoir généré le script de la base de données et de son utilisateur, j'ai débuté la rédaction de cet actuel compte rendu du projet au format PDF. J'ai d'abord créé le sommaire automatisé du compte rendu, puis ai écrit les différentes parties de ce dernier (rappel du contexte, rappel de la mission, présentation des étapes du projet et bilan final) en ajoutant les captures d'écran nécessaires.

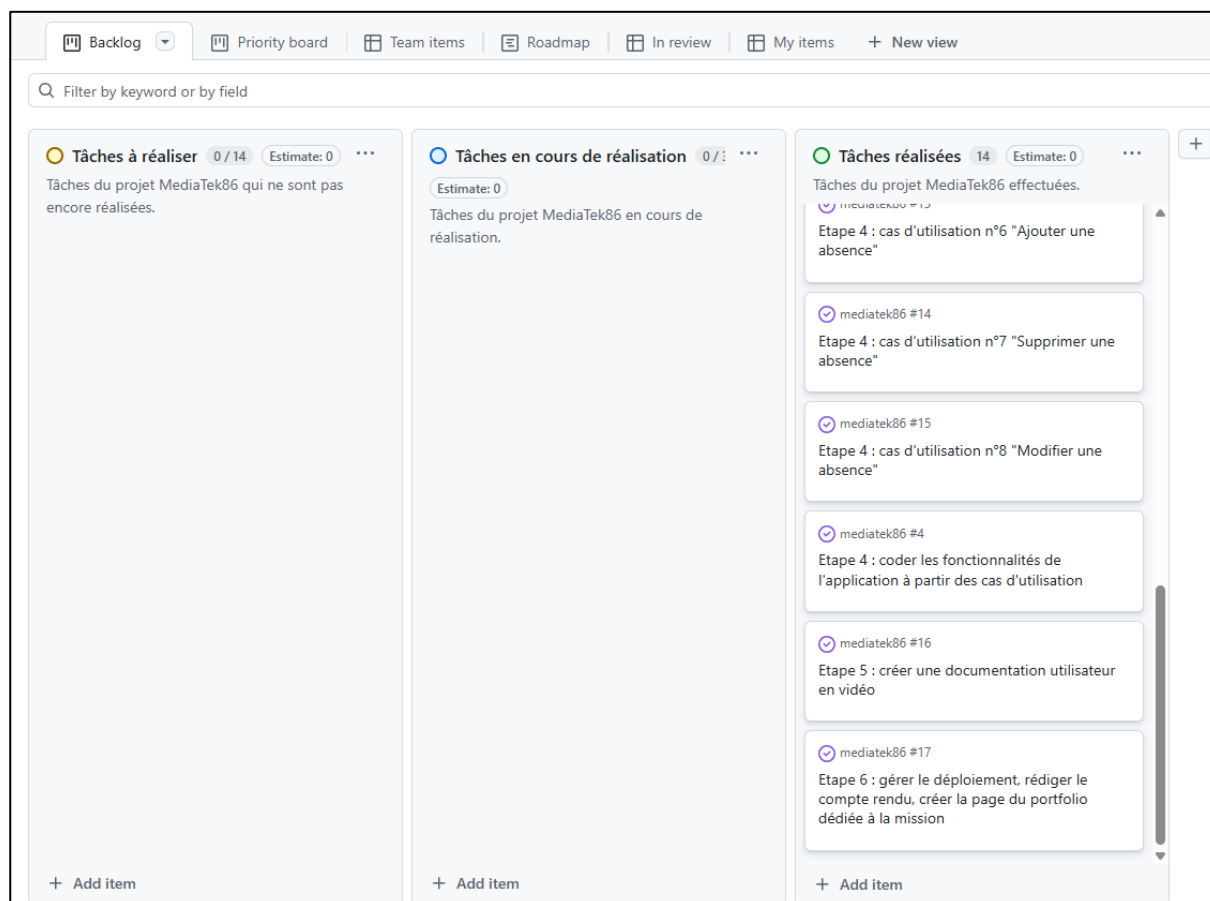
### 4. Création d'une page dédiée au projet dans le portfolio

Le compte rendu rédigé, j'ai créé la page du portfolio dédiée au projet, en commençant par rédiger une présentation concise du contexte et de la mission donnée. J'ai également ajouté un lien vers le dépôt distant GitHub de l'application qui contient les différentes étapes de sauvegarde et le script SQL complet de la BDD, ainsi que le lien vers l'actuel compte rendu d'activité. Enfin, j'ai intégré la vidéo de documentation utilisateur dans cette même page du portfolio, dans le but de présenter l'application et les possibilités qu'elle offre.

### 5. Bilan sur les objectifs atteints

Une fois cette dernière étape finie, l'installateur de l'application, le script complet de la base de données « mediatek86 » et de son utilisateur « mediatek86user », le compte rendu du projet au format PDF, ainsi que la page du portfolio réservée à ce même projet ont tous été complètement réalisés. Le projet et la mission confiée liée à celui-ci sont donc entièrement terminés.

Avancement du « project » (kanban) à ce stade du projet :



## Bilan final

Après réalisation de toutes les étapes présentées précédemment, l'application MediaTek86 était complètement finie et prête à l'utilisation. L'utilisateur responsable pouvait dès lors entièrement gérer le personnel des médiathèques du réseau MediaTek86 et ses informations, ainsi que ses absences grâce à l'application créée.

Par ailleurs, cette mission confiée et plus généralement ce projet dans sa totalité, m'ont permis de développer mes connaissances et compétences en développement d'applications et de solutions logicielles, de me familiariser avec des situations professionnelles réalistes, de me préparer davantage vis-à-vis de mes futures situations de stage en entreprise, de développer des compétences professionnelles, de communication et comportementales liées à l'exercice du métier, ainsi que d'alimenter mon parcours de professionnalisation retracé dans mon portfolio.