1. Mediatek86

1.1. Description

Une solution pour la médiathèque de Vienne

1.2. Avertissement

Ce projet est un projet scolaire réalisé dans le cadre de la formation BTS-SIO première année.
Il est fourni tel quel et n'est pas destiné à être utilisé en production.
L'application n'est pas conforme aux normes de sécurité et de qualité attendues pour une application professionnelle.
Elle est diffusée ici à titre d'exemple pour illustrer les compétences acquises par l'auteur dans le cadre de sa formation.
L'auteur ne fournit aucune garantie quant à son fonctionnement et ne pourra être tenu pour responsable de tout dommage causé par son utilisation.

1.3. Licence

Ce projet est sous licence MIT. Cela signifie que vous pouvez l'utiliser, le modifier et le distribuer comme bon vous semble, à condition de conserver la licence MIT dans les fichiers modifiés.

1.3.1. BddManager.cs

Le fichier BddManager.cs est une adaptation du code fourni par le professeur et n'est pas concerné par la licence MIT. Il reste la propriété de l'auteur original et ne doit pas être utilisé en dehors du cadre de ce projet.

1.4. Documentation du code source

Le code source est documenté en utilisant le format XMLDoc. Le compilateur C# génère automatiquement un fichier XML contenant la documentation du code source. Ce fichier est disponible dans le répertoire `bin\Debug` ou `bin\Release` après la compilation du projet.

1.4.1. SandCastle pour Visual Studio 2022

Conformément au cahier des charges, la documentation du code source est également générée avec SandCastle. Le projet SandCastle est disponible dans la solution sous le nom `Documentation`. Elle est déployée dans GitHub Pages à l'adresse suivante : Documentation

1.5. CI/CD

Le projet est configuré pour utiliser GitHub Actions pour la CI/CD.
Le workflow est défini dans le fichier `.github/workflows/ci.yml`.
La publication de la documentation est effectuée automatiquement à chaque push sur la branche `main` via le workflow `.github/workflows/static-pages.yaml` .

1.6. Architecture

Cette application est écrite en C# et utilise le framework .NET 8.0 accompagné de la couche interface utilisateur Microsoft WPF.
Ses données sont stockées dans une base de données MySQL installée sur le poste de l'utilisateur.
L'accès à la base de données se fait via Entity Framework Core et la classe d'accès BddManager.

1.7. Contexte de l'application

Dans le cadre de la formation BTS-SIO Première année il est proposé de réaliser une application imaginaire pour la médiathèque de Vienne.
Voici comment est présenté le projet.

Nous avons été intégré dans l'entreprise `ESN InfoTech Services 86` en tant que développeur junior.
L'entreprise nous a alors confié la réalisation d'une application Windows pour la médiathèque de Vienne.

Cette application doit permettre de gérer les absences du personnel dans une médiathèque.

L'application est mono-utilisateur et doit permettre de :

- Consulter la liste des employés
- Ajouter un employé
- Modifier un employé

- Consulter la liste des absences
- Ajouter une absence
- Modifier une absence
- Supprimer une absence

1.8. Fonctionnalités

Une étude du cachier des charges fourni par l'entreprise nous a permis de définir les fonctionnalités suivantes :

1.8.1. Contrôle d'accès à la base de données

La connexion entre la base de données et l'application est sécurisée par un contrôle d'accès. Un utilisateur de la base de données doit être créé avec les droits nécessaires pour accéder à la base de données. La chaîne de connexion est stockée dans le fichier `App.config` situé dans le répertoire de l'application. Notez que le fichier s'appelle `Mediatek86.dll.config` dans le répertoire `bin\Debug` ou `bin\Release` après la compilation du projet.

Voici par exemple une configuration valide pour la chaîne de connexion et les paramètres de runtime:

```
<?xml version="1.0" encoding="utf-8" ?>

<configuration>
    <configSections>
        <section name="system.data"
type="System.Data.Common.DbProviderFactoriesConfigurationSection, System.Data,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" />
        <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection,
EntityFramework, Version=6.0.0.0, Culture=neutral" requirePermission="false" />
    </configSections>
    <entityFramework
codeConfigurationType="MySql.Data.EntityFramework.MySqlEFConfiguration,
MySql.Data.EntityFramework">
        <defaultConnectionFactory
type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework"/>
        <providers>
            <provider invariantName="MySql.Data.MySqlClient"
                type="MySql.Data.MySqlClient.MySqlProviderServices,
MySql.Data.EntityFramework"/>
        </providers>
    </entityFramework>
    <connectionStrings>
        <add name="MyMediatek86DbContext"
             providerName="MySql.Data.MySqlClient"

connectionString="server=localhost;database=mediatek86;user=mediatek86;password=me
diatek86pwd" />
    </connectionStrings>
</configuration>
```

1.8.2. Contrôle d'accès à l'application

L'accès à l'application est sécurisé par un contrôle d'accès vérifiant les identifiants de l'utilisateur. Cette vérification est effectuée en comparant les identifiants saisis par l'utilisateur avec ceux stockés dans la base de données. Par souci de sécurité il a été convenu de ne pas stocker les mots de passe en clair dans la base de données. Les mots de passe sont donc hashés avant d'être stockés. Le hashage est effectué avec l'algorithme SHA256. Le hash est stocké sous forme de chaîne héxadécimale dans la base de données.

1.8.2.1. Dépannage de la connexion ou modification des paramètres de connexion

Il n'a pas été prévu dans le cahier des charges de l'application de permettre à l'utilisateur de modifier les paramètres de connexion à l'application'. Cependant, il est possible de modifier les paramètres de connexion en modifiant le contenu de la base de données. Pour ce faire, il est nécessaire de se connecter à la base de données avec un outil tel que MySQL Workbench. Il est alors possible de modifier les paramètres de connexion dans la table `responsable` de la base de données.

Voici comment créer l'utilisateur `admin` avec le mot de passe `adminpwd` :

```
INSERT INTO `responsable` (`login`, `pwd`) VALUES ('admin',  SHA2('adminpwd',
256));
```

Voici comment modifier le mot de passe de l'utilisateur `admin` pour le remplacer par `newpwd` :

```
UPDATE `responsable` SET `pwd` = SHA2('newpwd', 256) WHERE `login` = 'admin';
```

1.8.3. Gestion du personnel

L'application permet de consulter la liste des employés, d'ajouter un employé, de modifier un employé.
Le fonctionnement est simple; sur la page d'accueil de l'application, la liste des employés est affichée.
Un bouton `Ajouter` permet d'ajouter un employé.
Après avoir sélectionné un employé dans la liste, le bouton `Modifier` permet de modifier les informations de l'employé. Après avoir sélectionné un employé dans la liste, le bouton `Supprimer` permet de supprimer l'employé.

1.9. Processus de développement

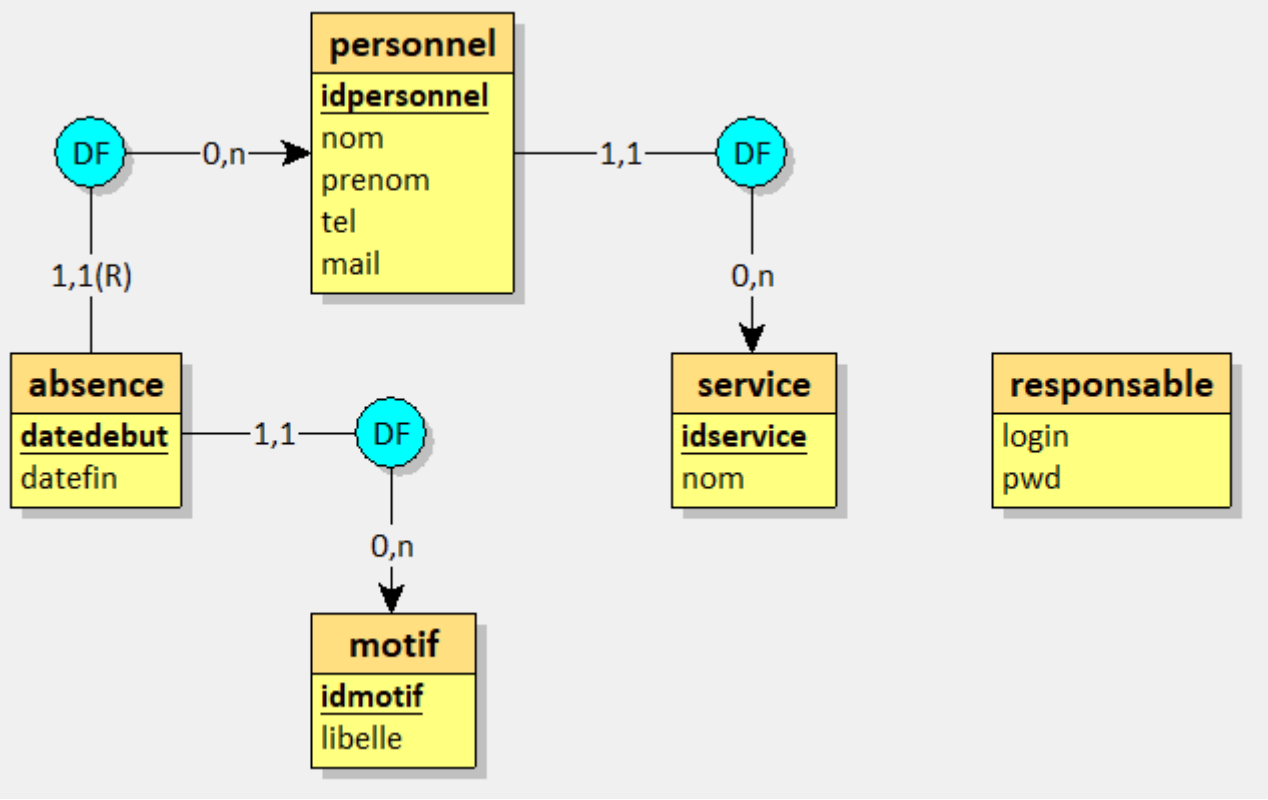
Le développement de l'application a été réalisé en plusieurs étapes.

1.9.1. Installation des outils de développement

- Installation de Visual Studio 2022
    - Les options C# et .NET 8.0 sont installées
- Installation de MariaDB 11.4.2 depuis le site officiel
- Installation de Looping MCD 4.0 depuis le site officiel

1.9.2. Conception de la base de données

Le modèle conceptuel de données a été réalisé avec Looping MCD 4.0.



La table `responsable` a été créée dans Looping puis le script de création des tables a été généré et a été complété pour créer la base de données appelée `mediatek86` et peupler les tables `motif` et `service` dans le même script.

Pour créer la base de données, il suffit de copier le contenu du script dans un éditeur de requêtes SQL tel que MySQL Workbench et de l'exécuter.

Une fois la base de données créée, nous avons créé un utilisateur `mediatek86` avec le mot de passe `mediatek86pwd` et les droits nécessaires pour accéder à la base de données. Cela a été fait avec les commandes suivantes :

```
CREATE USER 'mediatek86'@'localhost' IDENTIFIED BY 'mediatek86pwd';
GRANT ALL PRIVILEGES ON mediatek86.*
TO 'mediatek86'@'localhost';
```

1.9.2.1. Génération d'un jeu de données de test

Pour faciliter le développement de l'application, un jeu de données de test a été généré. Le jeu a été créé manuellement Ce jeu peut être modifié ou complété en fonction des besoins de test.
Il est consultable ici
Nous avons décider d'utiliser des sous-requêtes pour insérer les données dans les tables `personnel` et `absence` afin de garantir l'intégrité référentielle et de permettre de passer plusieurs fois le jeu de tests sans risquer de conflits d'identifiants.

1.9.2.2. Nettoyage de la base de données

Pour nettoyer la base de données et réinitialiser les données de test, il suffit d'exécuter les commande suivantes :

```
DELETE FROM `absence`;
DELETE FROM `personnel`;
```

1.9.3. Développement de l'application

1.9.3.1. Installation des extensions Visual Studio

Pour faciliter le développement, nous avons installé les extensions suivantes :

- SandCastle Help File Builder 2024.2.18.0

1.9.3.2. Création du projet

Une solution vide a été créée dans Visual Studio 2022. Le projet a été créé avec Visual Studio 2022 en utilisant le modèle `WPF App (.NET)`.
Le projet a été nommé `Mediatek86` et a été enregistré dans le répertoire `C:\Users\Briacl\source\repos\Mediatek86`.
Un projet supplémentaire a été ajouté pour la documentation du code source. Le projet a été nommé `Documentation` et a été enregistré dans le répertoire `C:\Users\Briacl\source\repos\Mediatek86\Documentation`.

1.9.3.2.1. Installation des packages NuGet

Pour faciliter le développement, nous avons installé les packages NuGet suivants :

- EntityFramework 6.4.4
- MySql.Data 8.4.0
- MySql.Data.EntityFramework 8.4.0

1.9.3.3. Contrôle du code source

Pour sécuriser le code source, nous avons créer un dépôt Git sur GitHub. Le code source est versionné et les modifications sont commentées pour faciliter la compréhension du code. Pour se conformer aux standards de GitHub, nous avons ajouté un fichier `README.md` à la racine du dépôt. Ce fichier contient une description du projet, une licence, une documentation du code source et des informations sur le processus de développement. Ce fichier est écrit en Markdown pour faciliter la lecture sur le site GitHub. Une version PDF est également disponible pour une lecture hors ligne.

Le dépôt a été initialisé avec un fichier `.gitignore` pour ignorer les fichiers temporaires et les fichiers de configuration de Visual Studio.

```
git init
git config --global user.email "briacl@cned"
git config --global user.name "briacl"
```

```
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin git@github.com:/briacl/Mediatek86.git
git push -u origin main
```

1.9.3.4. Développement des classes métier

Les classes métier ont été développées en suivant le modèle MVC.
Les classes `Personnel`, `Service`, `Motif`, `Responsable` et `Absence` ont été créées dans le répertoire Models.

1.9.3.4.1. Responsable

La classe `Responsable` est une classe métier qui représente un utilisateur de l'application.
Elle inclut une méthode `VerifierMotDePasse` qui permet de vérifier si le mot de passe saisi par l'utilisateur est correct.

1.9.3.4.2. MyDbContext

La classe `MyDbContext` est une classe qui hérite de `DbContext` et permet de faire correspondres les tables de la base de données avec les classes métiers. Elle met à profit les fonctionnalités d'Entity Framework et du connecteur natif Oracle MySql.Data pour simplifier l'accès aux données.
Nous avons choisi cette approche car c'est aujourd'hui la méthode la plus couramment utilisée pour accéder aux bases de données dans les applications .NET. Elle permet de simplifier le code en utilisant des classes métiers pour manipuler les données au lieu d'écrire des requêtes SQL à la main. Cette approche ORM (Object-Relational Mapping) permet de réduire les erreurs et de faciliter la maintenance du code. Cela permet également de bénéficier des fonctionnalités avancées d'Entity Framework telles que le suivi des modifications, les migrations de base de données et les requêtes LINQ.

Développement des vues et des contrôleurs

Les vues de l'application ont été créées en utilisant le designer de Visual Studio. Le choix du WPF a été fait pour sa facilité d'utilisation et sa compatibilité avec les applications Windows. De plus le WPF permet de créer des interfaces utilisateur riches et interactives. Même si l'application Mediatek86 est extrêmement simple, en théorie le WPF permet de créer des interfaces utilisateur modernes et ergonomiques.

Les vues ont été créées dans le répertoire Views . Chaque vue est un fichier XAML qui définit l'interface utilisateur de l'application. Elles sont associées à un contrôleur qui lui est associé. Les contrôleurs directement associés aux vues portent le même nom que la vue mais ont l'extension `.cs`. Les contrôleurs sont responsables de la logique métier de l'application. Ils interagissent avec les classes métier pour récupérer et enregistrer les données. Ils sont également responsables de la navigation entre les vues.

Le cahier des charges impose un confirmation de suppression ou de suppression pour éviter les suppressions accidentelles. Pour cela nous avons utilisé de simples MessageBox pour demander une confirmation à l'utilisateur. Par exemple pour la suppression d'un personnel :

```
```

```
Personnel? currentPersonnel = myDataGrid.SelectedItem as Personnel;
            if (currentPersonnel == null)
            {
                MessageBox.Show("Veuillez sélectionner un personnel à
supprimer.");
                return;
            }
```

1.9.3.4. Affichage de la liste des employés

La liste des employés est affichée dans un DataGrid. Le DataGrid est un contrôle WPF qui permet d'afficher des données sous forme de tableau. Il est très flexible et permet de personnaliser l'affichage des données. Dans notre cas, nous avons utilisé un DataGrid pour afficher les employés. Chaque ligne du DataGrid correspond à un employé. Les colonnes du DataGrid correspondent aux propriétés de l'employé. Par exemple, la colonne `Nom` affiche le nom de l'employé, la colonne `Prénom` affiche le prénom de l'employé, etc.