25/05/2020

Thomas Tissot

CFPT

Documentation Technique

Cadastre Parcelle mensuration

# Table des matières

[1. Table des matières 1](#_Toc41997602)

[2. Introduction 2](#_Toc41997603)

[3. Méthodologie 2](#_Toc41997604)

[4. Résumé du cahier des charges 2](#_Toc41997605)

[4.1 But de l’application 2](#_Toc41997606)

[4.2 Matériel et logiciel nécessaires 2](#_Toc41997607)

[5. Analyse fonctionnelle 2](#_Toc41997608)

[6. Analyse Organique 2](#_Toc41997609)

[6.1 Description détailler des méthodes 2](#_Toc41997610)

[Class ConnexionDb 2](#_Toc41997611)

[Classe UtilCsv 3](#_Toc41997612)

[Classe UtilDataGrid() 3](#_Toc41997613)

[Classe UtilDb() 3](#_Toc41997614)

[Classe UtilLog() 4](#_Toc41997615)

[7. Réalisation 4](#_Toc41997616)

[7.2 Arborescence du Projet 4](#_Toc41997617)

[8. Tests unitaires 4](#_Toc41997618)

# Introduction

Cette documentation a pour but de détailler en plusieurs partie mon application C# « Cadastre parcelle mensuration », réaliser dans le cadre du TPI (Travail pratique individuel) pour valider mon CFC d’informaticien.

Le premier jour de mon TPI j’ai reçu le cahier des charges de mon application par mail. Pour ma part je dois faire une application Windows Form qui permettra de visualiser, filtrer et modifier les données du cadastre importer avec un fichier CSV que mon application ajoutera dans la base de données. On peut également exporter les données de la base de données en fichier CSV.

# Méthodologie

J’ai choisi d’utiliser la méthodologie « WaterFall ». Celle-ci consiste une succession d’étapes, chacune d'entre elles menant à la suivante. Voici les 6 étapes de cette méthodologie :

* Exigences (analyse des besoins du mandant)
* Conception
* Planning
* Mise en œuvre
* Vérification
* Maintenance

Dans mon cas je ne vais pas utiliser la dernière étape dans le cadre mon projet.

# Résumé du cahier des charges

## 4.1 But de l’application

Le but de l’application est de permettre a l’utilisateur d’importer des données a partir d’un fichier csv et de visualiser ces données. Il sera aussi possible de filtrer et trié les données du tableau, l’utilisateur pourra exporter les données en fichier CSV tout en gardant les filtres. Il pourra également modifier certains champs.

## 4.2 Matériel et logiciel nécessaires

* Ordinateur (PC)
* Visual Studio 2019 – Langage C#
* GitHub Desktop
* Serveur EasyPHP

# Analyse fonctionnelle

Cette section va traiter la partie visible de l’application, donc ce que verra l’utilisateur final.

## Liste des fonctionnalités disponibles

* Importer un fichier CSV
* Exporter un fichier CSV
* Sélectionner des parcelle

# Analyse Organique

Cette partie de la documentation portera sur le fonctionnement interne de l’application, cette partie est vu exclusivement par le développeur. Pour que cette partie soit le plus clair possible je vais séparer les méthodes par class.

## 6.1 Description détailler des méthodes

### Class ConnexionDb

Cette classe permet d’ouvrir et de fermer la connexion avec la base de données.

Pour être instanciée elle a besoin de plusieurs paramètres :

* L’adresse du serveur (Dans mon cas 127.0.0.1)
* Le nom de la base de données
* Le nom de l’utilisateur
* Le mot de passe de l’utilisateur

#### Méthode OpenConnection()

Cette méthode publique a pour but d’ouvrir la connexion avec la base de données. Elle retourne un bool pour savoir si l’application a rencontré un bug lors de la connexion.

#### Méthode CloseConnection()

Cette méthode publique a pour but de fermer la connexion avec la base de données. Elle retourne un bool pour savoir si l’application a rencontré un bug lors de la déconnexion.

### Classe UtilCsv

Cette classe contient toutes les méthodes que j’utilise pour importer ou exporter des fichier CSV.

#### Méthode GetDataFromFile()

Cette méthode publique a pour but de créer une DataTable à partir d’un fichier CSV. Elle prend comme paramètre le chemin d’accès du fichier csv que l’utilisateur a sélectionné.

#### Méthode ExportDataToCSV()

Cette méthode publique a pour but de convertir une DataTable en fichier CSV. Elle prend comme paramètres une DataTable qui contient les informations des parcelles et un string qui contient le chemin d’accès où le fichier CSV va s’enregistrer.

### Classe UtilDataGrid()

Cette classe permet de gérer les DataGridView présente dans mon application.

#### Méthode ConvertDataGridViewToDataTable()

Cette méthode publique a pour but de convertir les parcelles présente dans mon DataGridView en DataTable.

A compléter

### Classe UtilDb()

Cette classe contient toutes les requêtes SQL.

#### Méthode GetParcelle()

Cette méthode publique a pour but de récupérer toute les parcelles présente dans la base de données et les convertis en type DataSet.

#### Méthode InsertParcelles()

Cette méthode publique a pour but de récupérer en paramètre une DataTable qui contient les parcelle importer par l’utilisateur et les insérer dans la base de données.

#### Méthode GetParcelleWithIds()

Cette méthode publique a pour but de récupérer en paramètre une liste d’ids qui correspond a chaque parcelle sélectionner par l’utilisateur dans le DataGridView. Ensuite cette méthode fait un select sur la base de données pour récupérer chaque parcelle correspondant au ids de la liste.

### Classe UtilLog()

Cette classe contient toutes les classes utiles à créer et écrire dans le fichier texte log a chaque import de CSV.

#### Méthode CreateLogFile()

Cette méthode a pour but de récupérer en paramètre une liste de string qui contient les erreurs lors de l’import ainsi qu’une variable de type int qui contient le nombre total d’erreur. Ensuite cette méthode créer un fichier texte dans le répertoire du projet et ajoute le nombre d’erreur ainsi que chaque erreur que l’import a rencontrée.

#### Méthode AddLineInLogFile()

Cette méthode a pour but de récupérer en paramètre un StreamWriter qui permettra d’écrire dans le fichier log ainsi qu’une variable de type string qui contient la ligne à écrire dans le fichier log.

# Réalisation

## 7.2 Arborescence du Projet

# Tests unitaires