EZCleanSoft - Place de marché

ITLightON

**Auteurs :**

Jérôme Perciot

**Résumé :**

Ce document décrit les échanges nécessaires pour la création du projet « OopWindowsFormsDesigner ».

**Visas :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Approuvé par** | **Date** | **Nom** | **Signature** |
| *JPT* | 06/05/2018 | PERCIOT Jérôme | *DigJPT#0001* |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Vérifié par** | **Date** | **Nom** | **Signature** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Historiques des éditions :**

|  |  |  |
| --- | --- | --- |
| **Édition** | **Date** | **Liste des paragraphes ajoutés ou modifiés** |
| 0\_1 | 08/01/2017 | Création du document |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table des matières

[2 Historique 5](#_Toc31705335)

[3 Concepts 5](#_Toc31705336)

[3.1 Fichier « UiKit » 5](#_Toc31705337)

[3.2 Fichier « UiKitProject » 5](#_Toc31705338)

[3.3 Architecture 5](#_Toc31705339)

[3.3.1 Assembly « OopDesigner » 5](#_Toc31705340)

# Historique

En 2020, j’ai pris la décision de développer mon propre système de design de composants / contrôles Windows Forms mais qui ne soit pas intégré à Visual Studio.

Le designer de Visual étant abandonné et pauvre depuis trop longtemps, mon système doit être exécuté en mode « Out-of-process » pour pouvoir être exécuté dans un environnement autonome.

Le but de cet outil est de prendre la place du designer de Visual Studio et de générer son contenu en utilisant les composants prévus à cet effet, et dans lequel vous pourrez décider de ce que vous voulez inscrire dans le designer de Visual Studio.

Un mode de compilation sera également proposé pour vous permettre de compiler le projet et de générer une assembly de test (sous réserve de faisabilité technique).

Un des nombreux avantages de cette solution, est que les applications Windows Forms pourront être portées sur .NET Core sans se préoccuper de cibler le .NET Framework pour profiter du design automatique.

# Concepts

## Fichier « UiKit »

Un nouveau type de fichier sera crée automatiquement par le système (stocké en dehors de la solution ou du projet sélectionné). Le type de ce fichier est un fichier « **Json** », sa lecture et son écriture seront donc très rapides.

## Fichier « UiKitProject »

Ce fichier décrit l’ensemble des fichiers « **UiKit** » qui ont été générés, et leur mode de fonctionnement en sortie.

Ce sont ces fichiers qui génèrent ensuite les « **.Designer.cs** » correspondants.

Un exemple de fichier UiKitProject :

{

Version : 1.0,

Date : 2020-02-04T13 :58 :13Z01,

Solution :

{

RootPath :…

}

}

## Architecture

### Assembly « OopDesigner »

Cette bibliothèque C# permet de gérer l’ossature de votre projet (« **.uiKitProject** »).

Cette librairie devra proposer les fonctionnalités suivantes :

* Lecture/écriture des fichiers Json de configuration. L’utilisation de **Newtonsoft.Json** sera préconisé.
* Interrogation des composants pouvant être ajoutés au designer interne. L’Api permet d’enregistrer si cela est nécessaire les composants d’autres parties, une fonction d’enregistrement sera alors prévue à cet effet.
* Les composantes de type « Designer » seront purement et simplement ignorées.