|  |
| --- |
|  |
| Syntaxi |
| Extracteur de documentation |
|  |
| **Thomas Auguey** |
| **22/10/2015** |

|  |
| --- |
| Ce guide définit une méthode de développement applicable aux langages de programmation orienté-objet et basé sur le patron Modèle-Vue-Contrôleur. |

SOMMAIRE

[Syntaxi 3](#_Toc434260999)

[Présentation 3](#_Toc434261000)

[Motivations 3](#_Toc434261001)

[Cahier des charges 3](#_Toc434261002)

[Eléments 3](#_Toc434261003)

[Système 4](#_Toc434261004)

[Méthodologie 5](#_Toc434261005)

[Conception 6](#_Toc434261006)

[Modèle de données 7](#_Toc434261007)

[Diagramme de classes 8](#_Toc434261008)

[Cas d’utilisations 9](#_Toc434261009)

[Réalisation 10](#_Toc434261010)

[Base de données 10](#_Toc434261011)

[Application cliente 12](#_Toc434261012)

# Syntaxi

## Présentation

Syntaxi est un gestionnaire de documentation capable d’extraire la documentation de fichiers textes dans des objets de données. Ces données sont ainsi exploitables par des applications tiers pour mettre en forme des documentations utilisateurs. L’intérêt du système repose sur la possibilité pour l’utilisateur final de créer ces règles d’extraction basées sur les expressions-régulières (bien connue des programmeurs en informatique). Le projet vise donc principalement les documentations techniques et les API informatiques.

## Motivations

L’avantage est de tirer parti de la flexibilité des expressions régulières. Contrairement aux logiciels sur le marché programmé pour analyser des langages spécifiques (C++,C#,Java,…) tel que Doxygen, Syntaxi prend la problématique dans le sens inverse en proposant de s’adapter au code existant en se basant sur des expression régulières pour extraire le texte intéressant.

## Cahier des charges

Le système devra proposer un modèle de données permettant de stocker des objet de contenu texte génériques et typés. Les objets seront associés à une expression régulière permettant leurs extractions depuis un fichier texte quelconque. Les objets ainsi créés serons stocké sous forme de données persistantes lisible par une application tiers.

Le système ne définissant pas de mise en forme des objets (classe de données, page de description, fonction, élément de template, …) l’utilisateur final devra typer ses expressions dans le but de les identifier lors de la mise en page.

Idéalement, l’application devra permettre de mettre à jour des projets existant et d’exporter ses données dans une base SQL distante.

L’application sera proposé sous forme de ligne de commande et d’IHM. L’interface en ligne de commande est utilisé dans un but d’automatisation du processus de génération des données.

## Eléments

### Projet

Contient les objets de données générées et les objets de syntaxe.

### Contenu

Société assimilé au groupe. Les informations connues comprennent le minimum nécessaire à l’entreprise (raison sociale)

### Syntaxe

Il s’agit d’une tache réalisé par un employé dans le cadre de son travail. Les informations importantes sont la nature de la prestation et la date d’intervention.

### Configuration

La configuration contient les informations de connexion au serveur de données. Cette configuration est unique à chaque HIM client installé.

## Système

Le système proposé comprendra :

* Un serveur de base de données SQL-Server 2012
* Une interface de gestion des données

# Méthodologie

La méthode de développement est basée sur un cycle en plusieurs temps analyse/modélisation/intégration/test/validation. Cette méthodologie permet d’assurer un développement par palier du système ou chaque cycle produit une application fonctionnelle.

1. Analyse du besoin
2. Modélisation UML et documentation des modifications
3. Intégration et développement technique
4. Tests unitaires et validation des modifications
5. Commit des modifications et incrémentation de la version du projet

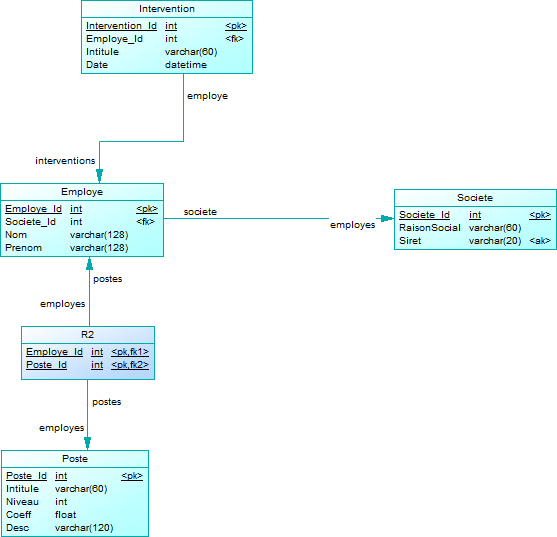
# Conception

Le système est modélisé sur une base UML comprenant :

* Un diagramme de données de la base SQL
* Un diagramme de classes du modèle de données applicatif
* Les cas d’utilisations HIM <> Base de données

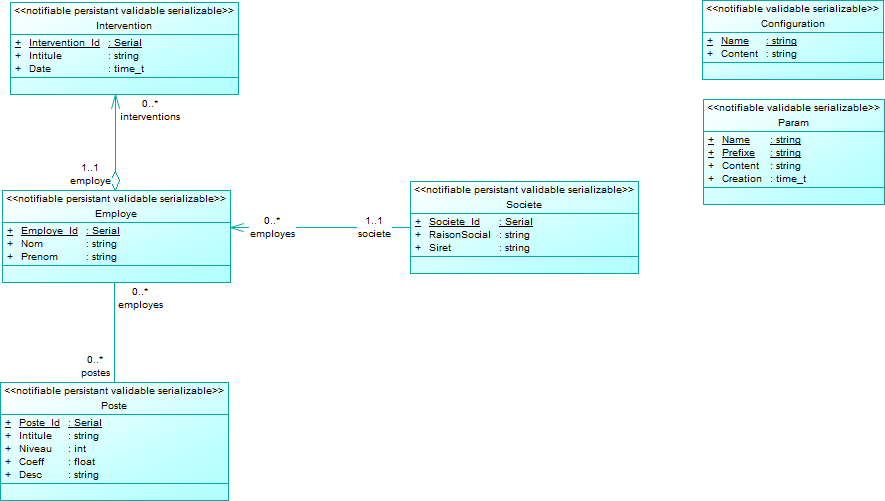
## Modèle de données

Modèle de la base de données SQL



## Diagramme de classes

Model de données de l’interface utilisateur



## Cas d’utilisations

# Réalisation

## Base de données

### Tables

Définition des tables SQL

|  |  |  |
| --- | --- | --- |
| **Employe**  Employé | | |
| Code | Data Type | Comment |
| Employe\_Id | int | Identificateur |
| Societe\_Id | int | Société |
| Nom | varchar(128) | Nom |
| Prenom | varchar(250) | Prénom |
| Index | Fields | Comment |
| PK\_EMPLOYE | Employe\_Id | Clé primaire |
| FK\_EMPLOYE\_R1\_SOCIETE | Societe(Societe\_Id) | Clé étrangère de la société |
| Trigger | Condition | Comment |
| - | - | - |

### Procédures stockées

Définition des procédures stockées SQL

## Application cliente

### Installation des packages NUGET (Visual Studio)

1. Ouvrir la console nuget (Outils > Gestionnaire de package nuget > Console du Gestionnaire de package nuget) :

> install-package extended.wpf.toolkit

> install-package WPFFolderBrowser

> install-package npgsql

1. Ouvrir le gestionnaire nuget (Outils > Gestionnaire de package nuget > Gérer les packages Nuget pour la solution…) :
   1. Pour chaque package cliquer sur Gérer et associer au projet WpfApp

### Entités

Définition des classes d’entités

|  |  |  |
| --- | --- | --- |
| **Employe**  Employé | | |
| Code | Data Type | Comment |
| Employe\_Id | int | Identificateur |
| Nom | string | Nom |
| Prenom | string | Prénom |
| Relation | Data Type | Comment |
| Societe | Societe | Société |