Application C# qui exploite un Web Service qui retourne un fichier XML

L'agglomération de Montpellier Méditerranée Métropole met à disposition de tout le monde un OpenData, c'est-à-dire un ensemble de ressources libres pour accéder à des informations répertoriées dans divers catégories (administration, citoyenneté, culture, économie, élection, …).

Nous allons nous intéresser aux stations de VéloMagg, les vélos en libre service mis à dispositions des usagers.

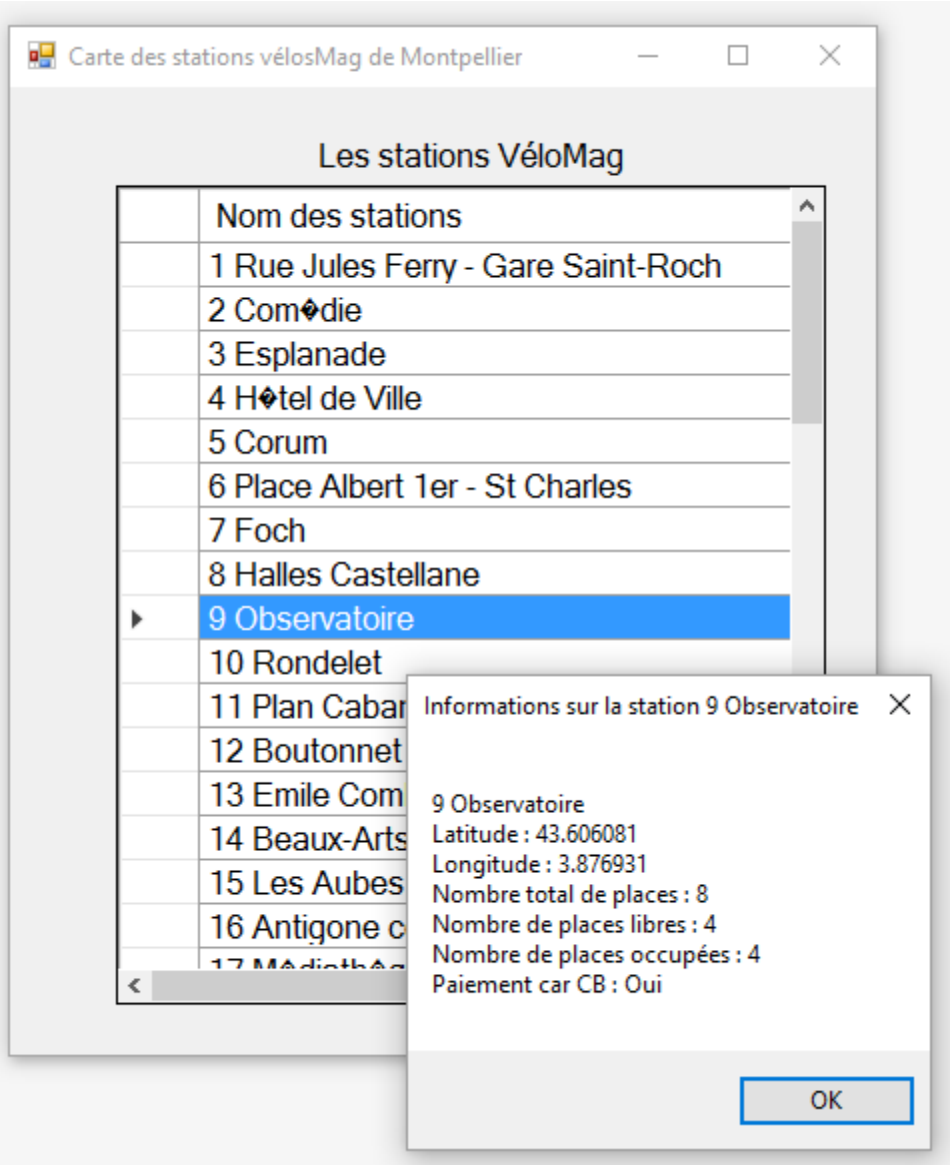
L'url
http://data.montpellier3m.fr/sites/default/files/ressources/TAM_MMM_VELOMAG.xml
renvoie un fichier XML qui contient divers renseignements sur les stations VéloMagg, et ceci en temps réel :

```
▼<vcs ver="1">
  ▼<sl>
     <si na="1 Rue Jules Ferry - Gare Saint-Roch" id="1" la="43.605366" lg="3.881346" av="7" fr="9" t
     <si na="2 Comédie" id="2" la="43.608148" lg="3.878778" av="11" fr="13" to="24" cb="1"/>
     <si na="3 Esplanade" id="3" la="43.609478" lg="3.881293" av="20" fr="12" to="32" cb="1"/>
     <si na="4 Hôtel de Ville" id="4" la="43.599088" lg="3.894866" av="7" fr="9" to="16" cb="1"/>
     <si na="5 Corum" id="5" la="43.613989" lg="3.881600" av="10" fr="6" to="16" cb="1"/>
     <si na="6 Place Albert 1er - St Charles" id="6" la="43.616768" lg="3.873375" av="18" fr="6" to='
     <si na="7 Foch" id="7" la="43.610989" lg="3.873345" av="3" fr="5" to="8"/>
     <si na="8 Halles Castellane" id="8" la="43.609935" lg="3.877208" av="7" fr="9" to="16"/>
     <si na="9 Observatoire" id="9" la="43.606081" lg="3.876931" av="4" fr="4" to="8" cb="1"/>
     <si na="10 Rondelet" id="10" la="43.603038" lg="3.875796" av="9" fr="7" to="16" cb="1"/>
     <si na="11 Plan Cabanes" id="11" la="43.608491" lg="3.868389" av="9" fr="3" to="12" cb="1"/>
     <si na="12 Boutonnet" id="12" la="43.622629" lg="3.868375" av="5" fr="7" to="12" cb="1"/>
     <si na="13 Emile Combes" id="13" la="43.616742" lg="3.879980" av="1" fr="7" to="8"/>
     <si na="14 Beaux-Arts" id="14" la="43.616698" lg="3.884981" av="6" fr="2" to="8"/>
     <si na="15 Les Aubes" id="15" la="43.618692" lg="3.893844" av="5" fr="3" to="8"/>
     <si na="16 Antigone centre" id="16" la="43.607942" lg="3.890634" av="9" fr="7" to="16" cb="1"/>
```

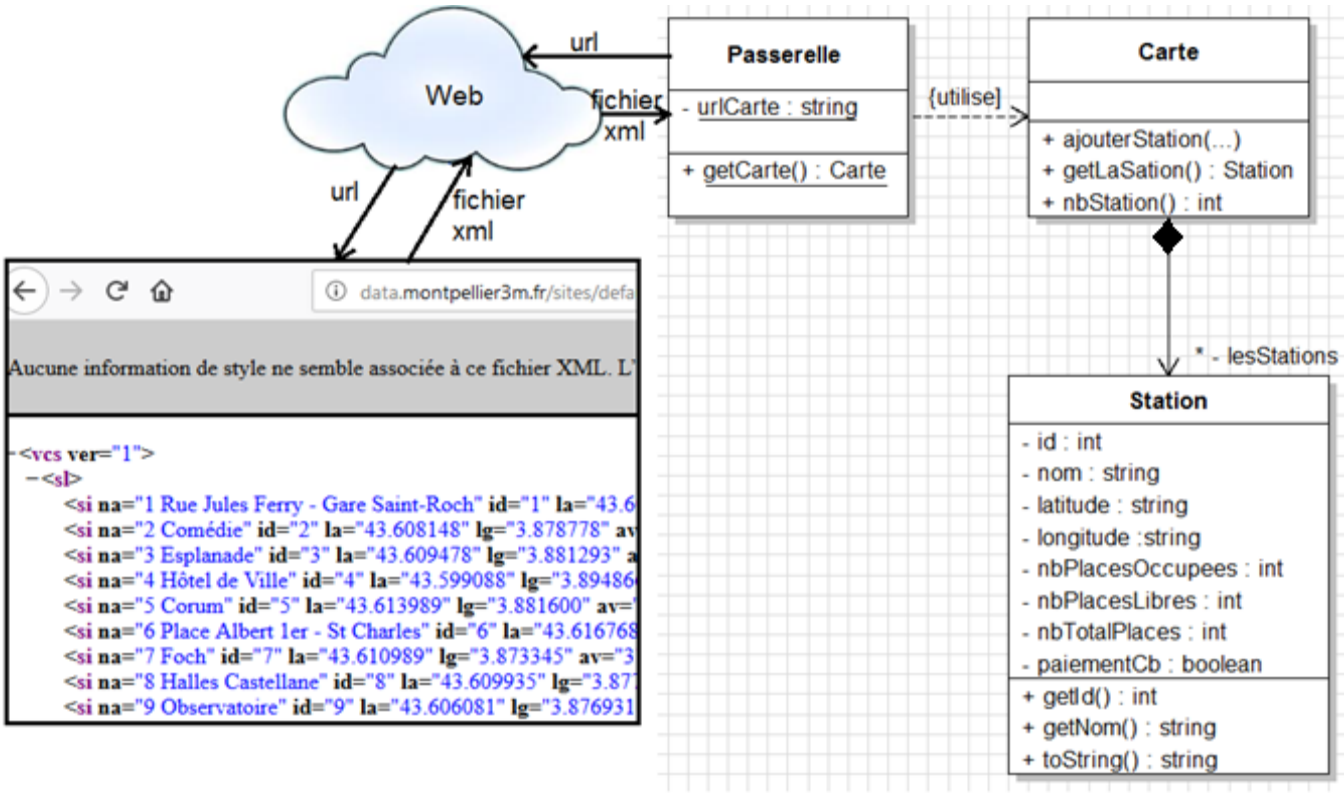
Chaque emplacement est décrit au sein d'une balise **<si>** par les attributs suivants :

- id : identifiant de la station
- na : nom de la station
- la : latitude
- lg : longitude
- av : nombre de places occupées
- fr : nombre de places libres
- to : nombre total de places
- cb : lecteur de carte bancaire disponible (0 pour non, 1 pour oui)

L'interface de l'application pourrait être la suivante, les noms des stations apparaissent dans un dataGridView. Sur click d'une station, on affiche le détail des informations.



Le schéma de fonctionnement et les classes utilisées :



Classes métier : Carte
Station

Classe technique : Passerelle

Précisions sur les méthodes de la classe Carte :

`ajouterStation (int id, string nom, …, boolean cb)`
→ Crée une station et la rajoute à la collection lesStations

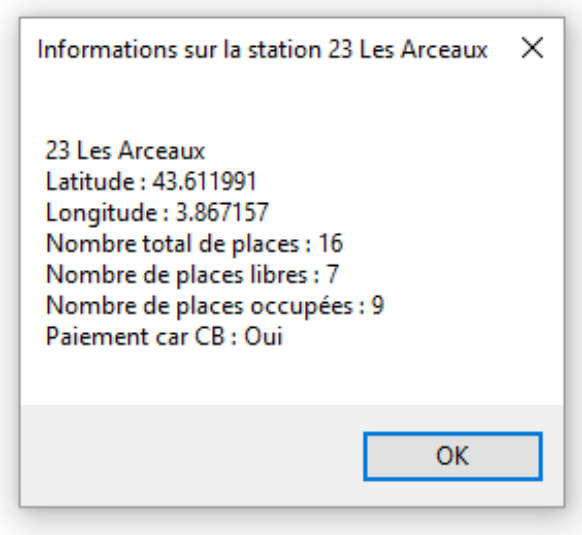
`getLaStation(int index) : Station`
→ Retourne la station située à l'index passé en paramètre dans la collection lesStations

`nbStation() : int`
→ Retourne le nombre de stations dans la collection lesStations

Exercice 1

Créer les classes Station et Carte

La méthode ToString() doit retourner une chaîne de caractères conforme à cet exemple :



Exercice 2

Pour faire le lien entre la ressource XML, l'interface graphique et les classes métiers, on va créer la classe technique Passerelle dont voici une partie du code :

```
namespace Velomagg
{
    class Passerelle
    {
        //Attributs statiques privés
        private static string urlCarte =
"http://data.montpellier3m.fr/sites/default/files/ressources/TAM_MMM_VELOMAG.xml";

        public static Carte getCarte()
        {
            try
            {
                //Créer une requête HTML
                HttpWebRequest requete = (HttpWebRequest)WebRequest.Create(urlCarte);
                requete.Method = WebRequestMethods.Http.Get;
                WebResponse rep = requete.GetResponse();

                //Déclarer un flux de données de type fichier
                StreamReader sr = new StreamReader(rep.GetResponseStream());

                //Préciser qu'il s'agit d'un document XML
                XmlTextReader xml = new XmlTextReader(sr);

                //Instanciation d'une carte
                Carte c = new Carte();

                //Tq il y a des lignes à lire
                while (xml.Read())
                {
                    //Si la ligne est une balise
                    if (xml.NodeType.Equals(XmlNodeType.Element))
                    {
                        //Si la balise est <si>
                        if (xml.LocalName.Equals("si"))
                        {
                            //on récupère les valeurs des différents attributs
                            string nom = xml.GetAttribute("na");
                            //****** code à compléter ici *******

                            c.ajouterStation(int.Parse(id), nom, latitude,longitude,
                            int.Parse(nbPlacesLibres), int.Parse(nbPlacesOccupees),
                            int.Parse(nbTotalPlaces), paiementCb);
                        }
                    }
                }
                return c;
            }
            catch (Exception ex)
            {
                Console.Write("Attention !! " + ex.Message);
                return null;
            }
        }
    }
}
```

Exercice 3 :

Créer l'interface graphique qui permet de remplir le dataGridView avec les différentes stations de la carte et programmer l'évènement **CellContentClick** du dataGridView

```
namespace Velomagg
{
    public partial class FrmVeloMagg : Form
    {
        //attribut privé
        private Carte laCarte;

        //Constructeur
        public FrmVeloMagg()
        {
            InitializeComponent();
        }

        //Permet d'instancier la carte à partir de la passerelle
        private void FrmVeloMagg_Load(object sender, EventArgs e)
        {
            //***** Code à compléter *******
        }

        //Permet d'afficher le detail d'une station
        private void dgvStations_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            //***** Code à compléter *******
        }
    }
}
```