

# UNIVERSITÉ DE MONTPELLIER

## M1 AIGLE

### Rapport : TP SOAP

**Étudiants :**

Denis BEAUGET Hayaat HEBIRET

**Année :** 2020 – 2021

**Encadrant :**

M. Abdelhak SERAI



UNIVERSITÉ  
DE MONTPELLIER

# Table des matières

<b>1 Objectifs :</b>	<b>3</b>
<b>2 Exercice 1 :</b>	<b>3</b>
2.1 SoapUI . . . . .	3
2.2 Postman . . . . .	3
2.3 Conclusion : . . . . .	4
<b>3 Exercice 2 :</b>	<b>4</b>
3.1 Code du service . . . . .	4
3.2 Résultat . . . . .	6
<b>4 Exercice 3 :</b>	<b>6</b>
4.1 Code du service : . . . . .	6
4.2 Résultat : . . . . .	7
<b>5 Exercice 4 :</b>	<b>7</b>
<b>6 Première version</b>	<b>8</b>
6.1 UML version non distribuée : . . . . .	8
6.2 Explications . . . . .	8
6.3 Résultat . . . . .	9
<b>7 Seconde version</b>	<b>10</b>
7.1 UML version distribuée . . . . .	10
7.2 Explications . . . . .	10
7.3 Résultat . . . . .	11
<b>Extrait de codes</b>	<b>13</b>

# 1 Objectifs :

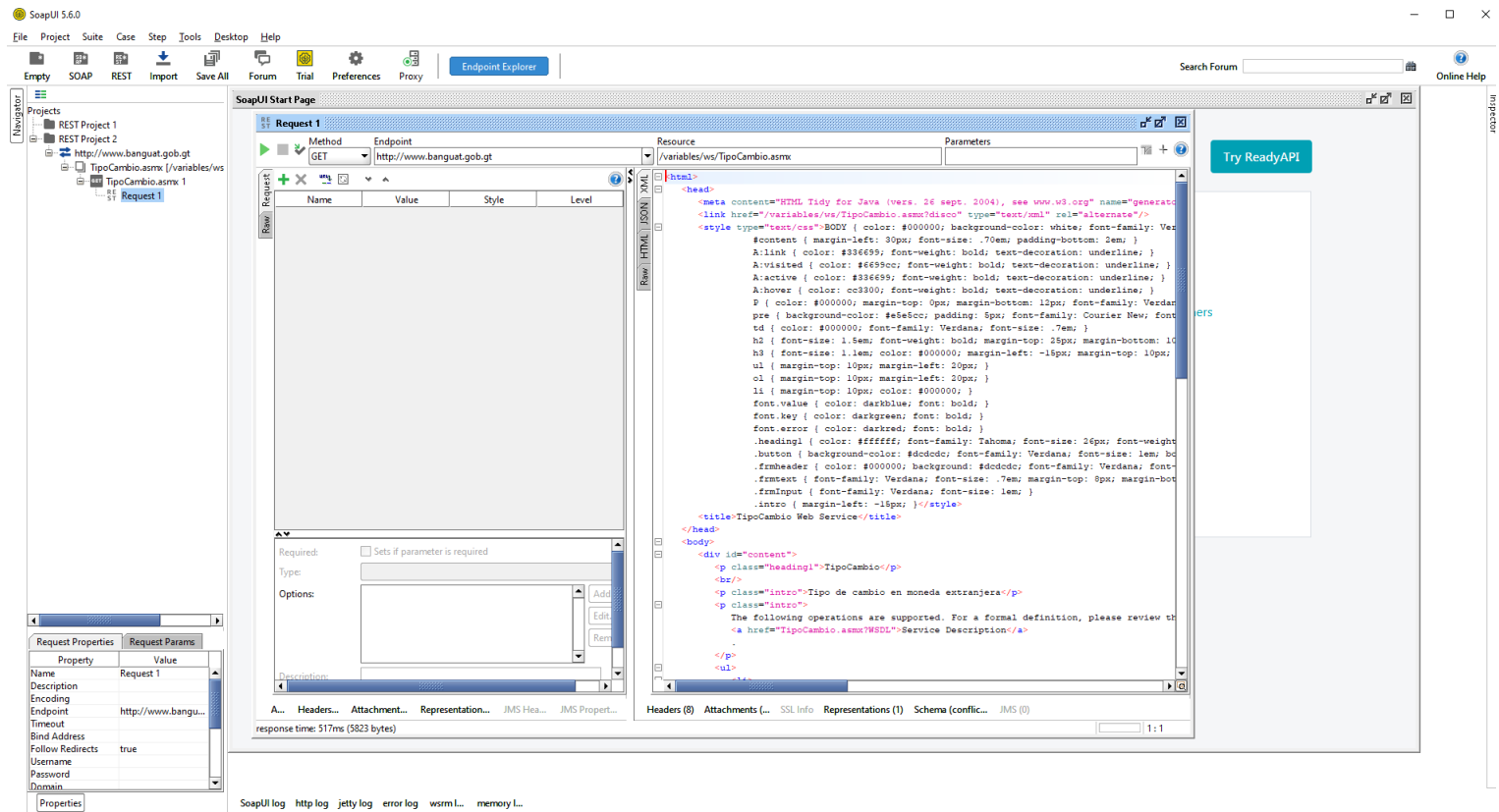
Ce TP sur les Service Web SOAP a pour objectif de nous faire comprendre les Services Web en C .NET à travers plusieurs exercices et outils. Avec par la suite la mise en pratique d'un exemple concret sur le thème de la gestion d'hôtels.

## 2 Exercice 1 :

### 2.1 SoapUI

SoapUI est un outil logiciel permettant de tester des services web REST/SOAP (plus globalement tout les services baser sur le protocole HTTP) à travers une interface logiciel.

*Exemple : test du service web : <http://www.banguat.gob.gt/variables/ws/TipoCambio.aspx> ?*

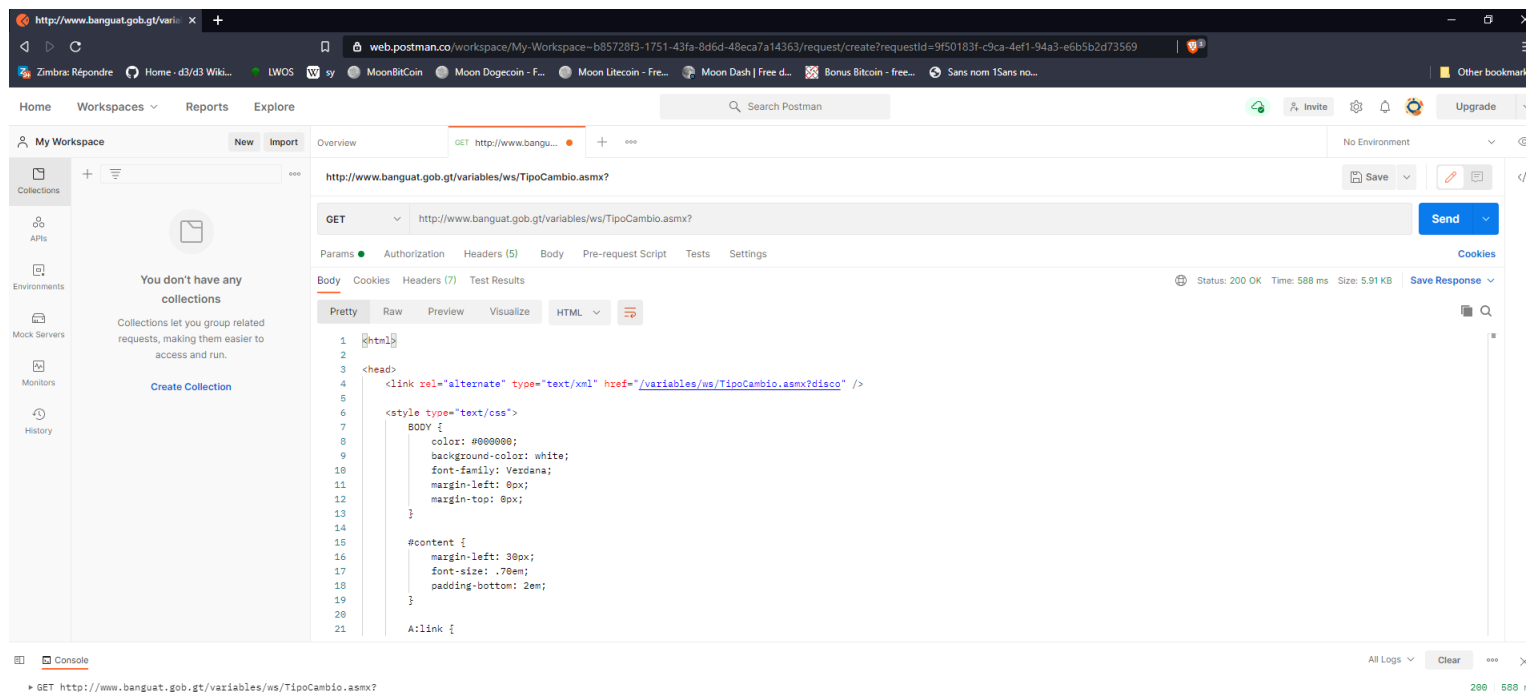


### 2.2 Postman

Postman est une application initialement une simple extension pour navigateur il s'est démocratisé et est passé en version standalone.

Il permet de tester les API et propose une version gratuite.

*Exemple : test du service web : <http://www.banguat.gob.gt/variables/ws/TipoCambio.aspx> ?*



## 2.3 Conclusion :

Ces 2 services sont relativement similaires (pour notre utilisation) même si SoapUI est plus "facile" d'accès à première vue. Néanmoins, les 2 articles cités mettent en avant d'autres différences :

- Les deux outils sont probablement les plus démocratisés, mais d'expérience SoapUI est "préférée" par les grandes entreprises et Postman est plutôt utilisé dans des entreprises plus petites.
- Le tarif : Postman est disponible à partir de 8\$ par mois soit 96\$ par année et comprend même des outils d'automatisation qui permettent de gagner du temps, alors que SoapUI est proposé à 659\$ par an.

Finalement, comme souvent, c'est au goût de chacun et des besoins des différents projets que l'un ou l'autre est utilisé.

## 3 Exercice 2 :

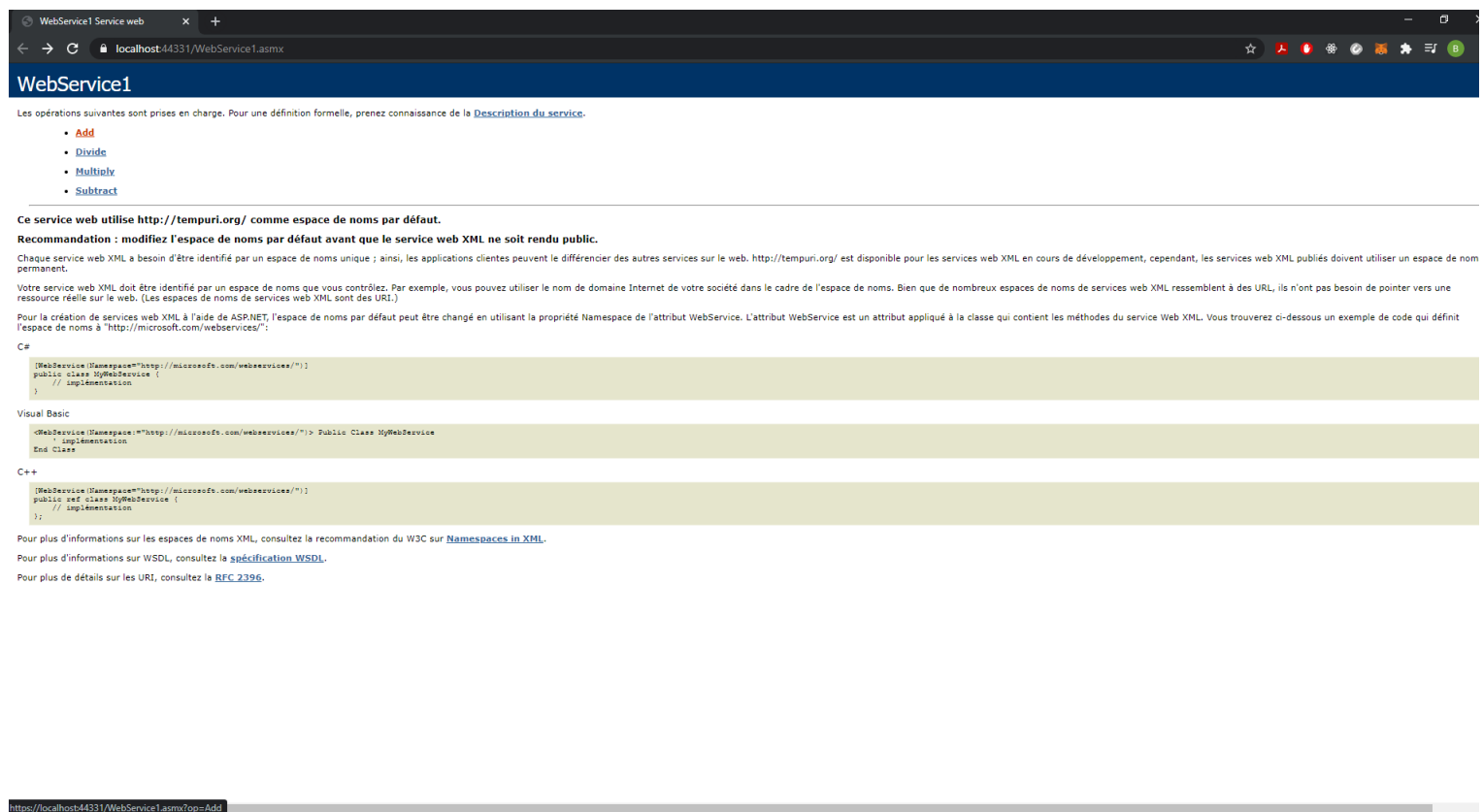
L'objectif de cet exercice est de créer un premier service web pour se familiariser avec la création et l'utilisation d'un service web.

Nous avons suivi ce tutoriel : <https://docs.microsoft.com/fr-fr/troubleshoot/dotnet/csharp/write-web-service>

### 3.1 Code du service

```
1 namespace MathServices
2 {
3     /// <summary>
4     /// Description résumée de WebService1
5     /// </summary>
6     [WebService(Namespace = "http://tempuri.org/")]
7     [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
8     [System.ComponentModel.ToolboxItem(false)]
9     // Pour autoriser l'appel de ce service Web depuis un script à l'aide d'ASP.NET AJAX,
10    supprimez les marques de commentaire de la ligne suivante.
11    [System.Web.Script.Services.ScriptService]
12    public class WebService1 : System.Web.Services.WebService
13    {
14
15        [WebMethod]
16        public int Add(int a, int b)
17        {
18            return (a + b);
19        }
20        [WebMethod]
21        public System.Single Subtract(System.Single A, System.Single B)
22        {
23            return (A - B);
24        }
25        [WebMethod]
26        public System.Single Multiply(System.Single A, System.Single B)
27        {
28            return A * B;
29        }
30        [WebMethod]
31        public System.Single Divide(System.Single A, System.Single B)
32        {
33            if (B == 0) return -1;
34            return Convert.ToSingle(A / B);
35        }
36    }
37 }
```

## 3.2 Résultat



WebService1

Les opérations suivantes sont prises en charge. Pour une définition formelle, prenez connaissance de la [Description du service](#).

- [Add](#)
- [Divide](#)
- [Multiply](#)
- [Subtract](#)

Ce service web utilise <http://tempuri.org/> comme espace de noms par défaut.

**Recommandation : modifiez l'espace de noms par défaut avant que le service web XML ne soit rendu public.**

Chaque service web XML a besoin d'être identifié par un espace de noms unique ; ainsi, les applications clientes peuvent le différencier des autres services sur le web. <http://tempuri.org/> est disponible pour les services web XML en cours de développement, cependant, les services web XML publiés doivent utiliser un espace de nom permanent.

Votre service web XML doit être identifié par un espace de noms que vous contrôlez. Par exemple, vous pouvez utiliser le nom de domaine Internet de votre société dans le cadre de l'espace de noms. Bien que de nombreux espaces de noms de services web XML ressemblent à des URL, ils n'ont pas besoin de pointer vers une ressource réelle sur le web. (Les espaces de noms de services web XML sont des URI.)

Pour la création de services web XML à l'aide de ASP.NET, l'espace de noms par défaut peut être changé en utilisant la propriété Namespace de l'attribut WebService. L'attribut WebService est un attribut appliqué à la classe qui contient les méthodes du service Web XML. Vous trouverez ci-dessous un exemple de code qui définit l'espace de noms à '<http://microsoft.com/webservices/>' :

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // Implementation
}
```

Visual Basic

```
<WebService(Namespace="http://microsoft.com/webservices/")> Public Class MyWebService
    ' Implementation
End Class
```

C++

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public ref class MyWebService {
    // Implementation
};
```

Pour plus d'informations sur les espaces de noms XML, consultez la recommandation du W3C sur [Namespaces in XML](#).

Pour plus d'informations sur WSDL, consultez la [spécification WSDL](#).

Pour plus de détails sur les URI, consultez la [RFC 2396](#).

<https://localhost:44331/WebService1.asmx?op=Add>

## 4 Exercice 3 :

L'objectif de cet exercice est d'importer un service web existant et de le consommer dans une application console classique.

### 4.1 Code du service :

```
1 namespace ConsommationWebService
2 {
3     class Program
4     {
5         static void Main(string[] args)
6         {
7             ServiceCalculatrice.CalculatorSoapClient test = new ServiceCalculatrice.
8             CalculatorSoapClient();
9             Console.WriteLine("Test addition :");
10            int res = test.Add(6, 9);
11            Console.WriteLine(res);
12            Console.WriteLine("Test division :");
13            res = test.Divide(6, 9);
14            Console.WriteLine(res);
15            Console.WriteLine("Test multiplication :");
16            res = test.Multiply(6, 9);
17            Console.WriteLine(res);
18            Console.WriteLine("Test soustraction :");
19            res = test.Subtract(6, 9);
20            Console.WriteLine(res);
21        }
22    }
23 }
```

```

21         Console.ReadLine();
22
23     }
24 }
25

```

## 4.2 Résultat :

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsommationWebService
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             ServiceCalculatrice.CalculatorSoapClient test = new ServiceCalculatrice.CalculatorSoapClient();
14             Console.WriteLine("Test addition :");
15             int res = test.Add(6, 9);
16             Console.WriteLine(res);
17             Console.WriteLine("Test division :");
18             res = test.Divide(6, 9);
19             Console.WriteLine(res);
20             Console.WriteLine("Test multiplication :");
21             res = test.Multiply(6, 9);
22             Console.WriteLine(res);
23             Console.WriteLine("Test soustraction :");
24             res = test.Subtract(6, 9);
25             Console.WriteLine(res);
26             Console.ReadLine();
27         }
28     }
29 }
30
31

```

```

Test addition :
15
Test division :
1
Test multiplication :
54
Test soustraction :
-3

```

## 5 Exercice 4 :

Git de l'application : <https://github.com/Beauget/TPServiceWebSOAP>

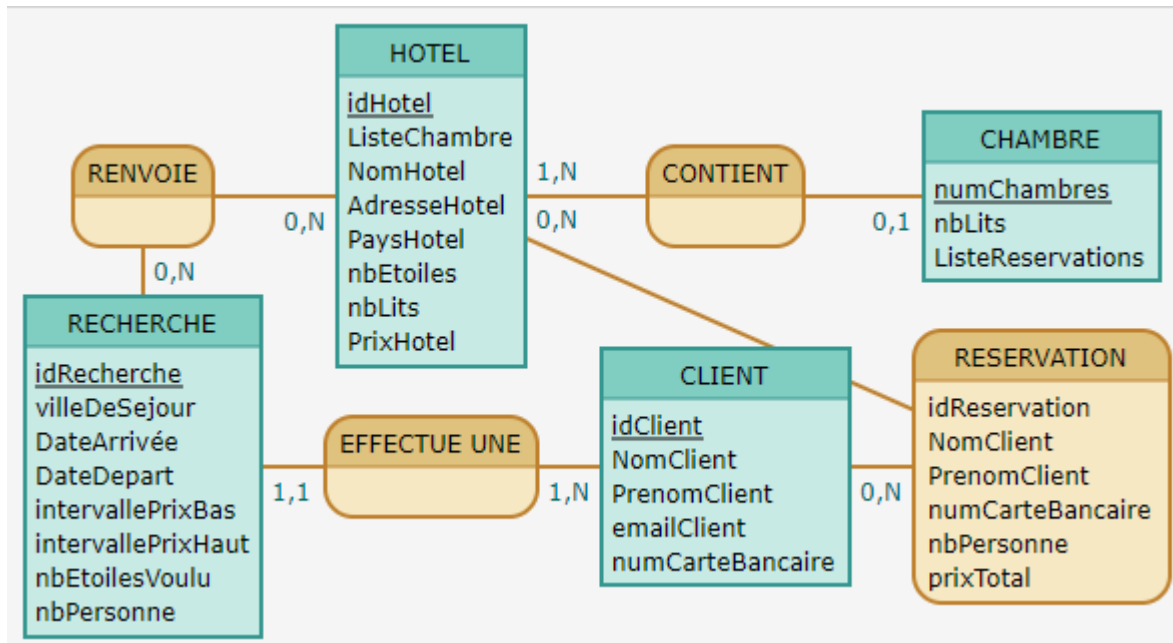
Cette application propose une interface permettant à un utilisateur de saisir les informations suivantes : Une ville de séjour, une date d'arrivée, une date de départ, un intervalle de prix souhaité, une catégorie d'hôtel : nombre d'étoiles, le nombre de personnes à héberger. En réponse, l'application lui retourne une liste d'hôtels qui répondent à ses critères et où pour chaque hôtel les informations suivantes sont affichées : nom de l'hôtel, adresse de l'hôtel (pays, ville, rue, numéro, lieu-dit, position GPS), le prix à payer, nombre d'étoile, nombre de lits proposés. L'utilisateur choisira un hôtel dans la liste proposée et l'application lui demandera les informations suivantes : le nom et prénom de la personne principale à héberger, les informations de la carte de crédit de paiement. L'application utilisera ces informations pour réaliser la réservation de l'hôtel en question.

### 2 versions sont attendues :

- La première sera non distribuée.
- La seconde sera en version distribuée (à l'aide de service web SOAP) avec l'ajout d'agence de voyages en intermédiaire entre les hotels et les clients.

## 6 Première version

### 6.1 UML version non distribuée :



### 6.2 Explications

Cette version est plutôt classique dans son fonctionnement, elle permet de poser la logique métier nécessaire pour les 2 versions de l'application et propose une interface console avec plusieurs commandes permettant d'interagir avec l'application et tester les fonctionnalités disponibles.

Pour tester notre application nous avons créés plusieurs hotels ainsi qu'une petite fonction pour initialiser les chambres de manière randomisé.

```

1 public void InitChambre()
2 {
3     var rand = new Random();
4     int numChambre = 0;
5     int x = rand.Next(10, 40);
6     for(int i = 0; i < x; i++)
7     {
8         int lits = rand.Next(1, 4);
9         Chambre chambre = new Chambre(numChambre, lits);
10        numChambre += 1;
11        this.ListChambres.Add(chambre);
12    }
13 }
14

```

#### Commandes disponible :

- **Faire une recherche** suivant plusieurs critères :  
le pays de l'hôtel, son nombre d'étoiles, l'intervalle de prix, le nombre de personnes et la date du séjour.

En réponse l'application affiche une liste d'hôtel répondant à **tout** les critères du client.



### - Faire une réservation :

L'application permet de faire une réservation dans la liste d'hôtel **précédemment renvoyé dans la recherche** (ceci est un choix de conception, soit un client fait forcément une recherche avant de passer une réservation ou alors toute la liste de base des hôtels sera utilisé pour la réservation)

### - Afficher tout les hôtels de l'application :

Tout simplement afficher tout les hôtels de l'application si le client n'a pas de critères particuliers.

## 6.3 Résultat

```
Saisir le chiffre correspondant à la fonctionnalité désiré :
0 : Quittez l'application
1 : Faire une recherche
2 : Sélectionnez votre Hotel et passez réservation
```

Figure 1 : Menu de l'application

```
LogiqueMetierHotel.Hotel
- Ibis
- 2220 Avenue du pigeon
- Montpellier
- France
- 2
- 45
Numéro : 0 | Nombre de lits : 3
Numéro : 1 | Nombre de lits : 1
Numéro : 2 | Nombre de lits : 2
Numéro : 3 | Nombre de lits : 1
Numéro : 4 | Nombre de lits : 1
Numéro : 5 | Nombre de lits : 2
Numéro : 6 | Nombre de lits : 1
Numéro : 7 | Nombre de lits : 2
Numéro : 8 | Nombre de lits : 3
Numéro : 9 | Nombre de lits : 3
Numéro : 10 | Nombre de lits : 2
Numéro : 11 | Nombre de lits : 1
Numéro : 12 | Nombre de lits : 2
Numéro : 13 | Nombre de lits : 2
Numéro : 14 | Nombre de lits : 1
Numéro : 15 | Nombre de lits : 1
Numéro : 16 | Nombre de lits : 2
Numéro : 17 | Nombre de lits : 2
Numéro : 18 | Nombre de lits : 1
Numéro : 19 | Nombre de lits : 3
Numéro : 20 | Nombre de lits : 1
Numéro : 21 | Nombre de lits : 1
Numéro : 22 | Nombre de lits : 2
Numéro : 23 | Nombre de lits : 1
Numéro : 24 | Nombre de lits : 1
```

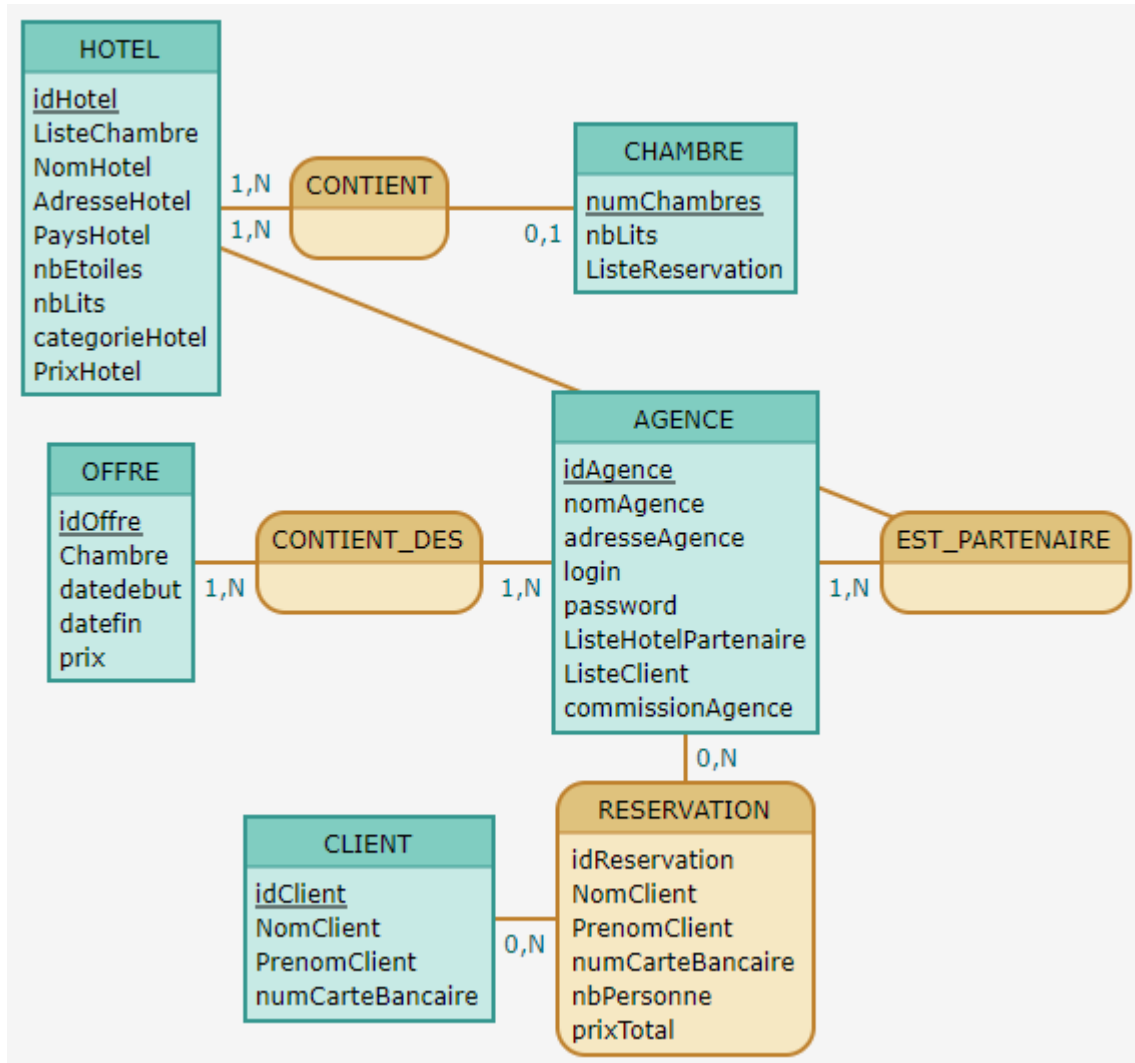
Figure 2 : Initialisation des chambres

```
Veuillez indiquer le nom de l'hotel que vous voulez sélectionner
BipCoyote
Veuillez indiquer les informations suivantes pour que votre réservation soit prise en compte : Nom, Prénom, Numéro de carte bancaire, Nombre de personnes, Date d'arrivé et de départ
Beauget
Denis
1345532
3
07/04/2021
13/04/2021
6
LogiqueMetierHotel.Hotel
- BipCoyote
- 700 Chemin du lapin
- Madrid
- Espagne
- 1
- 19
Numéro : 2 | Nombre de lits : 3
Beauget
- Denis
- 07/04/2021 00:00:00
- 13/04/2021 00:00:00
- Prix total de votre réservation : 342
Votre réservation a été effectué, bonne journée et à bientôt
```

Figure 3 : Procéssus de réservation

## 7 Seconde version

### 7.1 UML version distribuée



### 7.2 Explications

Cette version réutilise une partie des éléments métier réalisé précédemment mais nous avons effectué quelque changements de conception pour permettre l'ajout de nouveaux éléments (les offres et les agences).

Pour implémenté cette version nous avons décider de mettre la logique métier et l'implémentation de notre service web dans une première solution (**correspondant à l'hôtel, il suffit donc de créer un nouveau service pour ajouter un hotel au service**) et la consommation de notre service avec l'interface (**correspondant à une agence suivant le login et mot de passe utilisé**) dans une autre solution console.

Cette application propose les mêmes fonctionnalités que la précédente mais avec l'utilisation d'un service web et l'implémentation d'agence partenaire qui propose des offres au client qui se connecte à l'application.

**Commandes disponible :**

- **Afficher les offres (sans images)**

L'application renvoie la liste d'offre (provenant du service web) sans image et avec les différentes informations sur les offres.

- **Afficher les offres (avec images)**

L'application renvoie la liste d'offre (provenant du service web) et avec les différentes informations sur les offres en plus les images correspondant aux chambres proposés dans les offres sont télécharger dans un dossier de l'application.

**Disclaimer :** Nous avons utilisé des images qui se télécharge sur l'ordinateur de l'utilisateur, suivant le path indiqué dans l'application de l'agence, cette fonctionnalité ne marchera pas si le PATH n'est pas modifié.

```

1  public class HotelDisponibilite : System.Web.Services.WebService
2  {
3      private IFormatProvider culture = new CultureInfo("en-US", false);
4
5      public static string path = "C:\\Users\\beaug\\Desktop\\M1S2\\ArchiDistrib\\assets\\";

```

#### - Afficher les offres selon vos dates (avec GUI)

L'application affiche les offres rentrant dans les critères temporelle indiqué par l'utilisateur avec les différentes informations sur les offres avec une GUI permettant de voir les informations relative aux offres et les photos des chambres directement dans une fenêtre et de pouvoir réserver directement via l'application grâce à un formulaire (voir photos).

#### - Afficher les offres (avec GUI)

L'application renvoie la liste d'offre (provenant du service web) avec les différentes informations sur les offres avec une GUI permettant de voir les informations relative aux offres et les photos des chambres directement dans une fenêtre et de pouvoir réserver directement via l'application grâce à un formulaire (voir photos).

#### - Faire une réservation :

L'application permet de faire une réservation parmi la liste d'offres proposé par l'agence dans l'hotel partenaire.

## 7.3 Résultat

```

Saisir le chiffre correspondant à la fonctionnalité désiré :
0 : Quittez l'application
1 : Afficher les offres disponibles de notre agence (sans image)
2 : Afficher les offres disponible de notre agence (avec image téléchargez : vérifiez le PATH)
3 : Afficher les offres disponible de notre agence selon vos dates (avec GUI)
4 : Afficher toutes les offres disponible (avec GUI)
5 : Sélectionnez votre offre et passez réservation

```

Figure 1 : Menu de l'application

```

Bienvenue dans notre espace agence !
Saisir le chiffre correspondant à la fonctionnalité désiré :
0 : Quittez l'application
1 : Afficher les offres disponibles de notre agence (sans image)
2 : Afficher les offres disponible de notre agence (avec image téléchargez : vérifiez le PATH)
3 : Afficher les offres disponible de notre agence selon vos dates (avec GUI)
4 : Afficher toutes les offres disponible (avec GUI)
5 : Sélectionnez votre offre et passez réservation
1
Entrez le nombre de personnes :
2
Entrez la date de debut au format dd/mm/yyyy :
03/04/2021
Entrez la date de fin au format dd/mm/yyyy :
10/04/2021
- Id Offre (pour votre réservation) : IbisBudget-1
- Numéro Chambre et nombre de lits : 0 | 2 lits
-Prix Total : 294

```

Figure 2 : Processus d'affichage d'une offre

Agence

Id de l'offre : IbisBudget-4

Photo de la chambre (ci-dessous) et Nombre de lits : 4 lits

Informations

Nom de l'hotel : IbisBudget  
 Adresse : 230 Avenue des roses  
 Etoiles : 2  
 Prix nuit/personne : 35

Prix total de l'offre : 168€

Date de disponibilité :  
 Arriver : 09/04/2021  
 Départ : 13/04/2021

Réservation

Cette offre vous plaît ?  
 Remplissez notre formulaire  
 de réservation :

IdOffre

Nom

Prénom

Nb Personnes

Numéro CB

Figure 3 : Affichage via l'interface graphique

Agence

Id de l'offre : IbisBudget-1

Photo de la chambre (ci-dessous) et Nombre de lits : 2 lits

**Informations**

Nom de l'hôtel : IbisBudget  
Adresse : 230 Avenue des roses  
Etoiles : 2  
Prix nuit/personne : 35

Prix total de l'offre : 294€

Date de disponibilité :  
Arriver : 03/04/2021  
Départ : 10/04/2021

**Réservation**

Cette offre vous plaît ?  
Remplissez notre formulaire  
de réservation :

IdOffre :

Nom :

Prénom :

Nb Personnes :

Numéro CB :

**Photo de la chambre**

Nom de réservation : beauget denis  
- Date d'arrivée : 03/04/2021  
- Date de départ : 10/04/2021  
- Prix total de votre réservation : 294




Figure 4 : Après réservation

## Extrait de codes

Ces extraits de code sont des briques logiciel que nous avons souhaité intégrer dans le rapport, elles regroupent souvent plusieurs appels à des fonctions externe. L'objectif est de montrer nos fonctions "synthèse" mais qui sont le "cœur" de notre application (Afficher des offres/Passer une réservation) sans pour autant intégrer l'intégralité du code.

Cela est censé permettre au lecteur une compréhension global de l'application.

```

1  public List<Offre> createOffre()
2      {
3          List<Offre> testList = new List<Offre>();
4          Offre offreTemp = new Offre();
5          float prix;
6          Random r = new Random();
7          DateTime deb = DateTime.ParseExact("03/04/2021", "dd/MM/yyyy", culture);
8          DateTime fin = DateTime.ParseExact("10/04/2021", "dd/MM/yyyy", culture);
9
10
11         foreach (TypeChambre i in hotelPasCher.ListChambres)
12         {
13             prix = hotelPasCher.prixNuit +
14                 (hotelPasCher.prixNuit * agenceChoisis.commissionAgence);
15             offreTemp = new Offre(hotelPasCher.nomHotel + r.Next(40), i, deb, fin, prix);
16             testList.Add(offreTemp);
17         }
18
19         return testList;
20
21     }
22

```

#### Création d'offre version distribué

```

1  public void reservationHotel(List<Hotel> resList, List<Hotel> baseList)
2      {
3
4          foreach (Hotel x in baseList)
5          {
6              if (x.nomHotel.Equals(nom))
7              {
8                  TimeSpan total = (dateDepart - dateArrivee);
9                  double temp = total.TotalDays;
10                 double prixTotal = (int)(x.prixNuit * nbPersonne) * temp;
11                 Reservation res = new Reservation(nomPersonne, prenom, numeroCarte,
12                     dateArrivee, dateDepart, nbPersonne, prixTotal);
13                 //recherche la premiere chambre libre et fais la reservation
14                 //si elle n'existe pas/la reservation n'a pas pu etre effectuer renvoie null
15                 TypeChambre chambre = x.Reserver(res);
16
17                 if (chambre.Equals(z)==false)
18                 {
19                     //info hotel
20                     Console.WriteLine(x.ToString());
21                     //info chambre
22                     Console.WriteLine(chambre.ToString());
23                     //info reservation
24                     chambre.ToStringListReservation();
25                 }
26             }
27         }
28     }
29

```

```

25         Console.WriteLine("Votre réservation a été effectué");
26     }
27     else
28     {
29         Console.WriteLine("Désoler il n'y a pas de chambre disponible dans
30         cet hotel pour vous");
31     }
32 }
33 }
34 }

```

### Extrait réservation

```

1  //Afficher les offres disponible
2  [WebMethod (EnableSession =true)]
3  public List<Offre> AfficherOffreDisponible(string login, string password,
4  string dateArrive, string dateDepart,int nbPersonne)
5  {
6
7      DateTime dt1 = DateTime.ParseExact(dateArrive, "dd/MM/yyyy", culture);
8      DateTime dt2 = DateTime.ParseExact(dateDepart, "dd/MM/yyyy", culture);
9      this.agenceChoisis = checkConnexion(login, password);
10     List<Offre> Listres = new List<Offre>();
11     if (this.agenceChoisis != null)
12     {
13
14         Listres = listTemp;
15
16     }
17     else
18     {
19         Console.WriteLine("Désoler votre identification a échoué ! ");
20     }
21
22     return Listres;
23 }

```

### Web méthode d'affichage des offres (sans images)

```

1  [WebMethod(EnableSession = true)]
2  public Reservation faireReservation(string login, string password,
3  string idOffre, string nomPersonne, string prenom, int numeroCB, int nbPersonne)
4  {
5
6
7      checkConnexion(login, password);
8      Reservation resFinal = new Reservation();
9
10     foreach (Offre x in listTemp)
11     {
12
13         if (x.idOffre == idOffre)
14         {
15
16             Reservation res = new Reservation(nomPersonne, prenom, numeroCB,

```

```

17         x.deb, x.fin, nbPersonne, x.prixTotalOffre);
18         resFinal = res;
19         Client temp = new Client(nomPersonne, prenom, numeroCB);
20         agenceChoisis.ClientAgence.Add(temp);
21         TypeChambre chambre = hotelPasCher.Reserver(res);
22         return resFinal;
23     }
24 }
25 return resFinal;
26 }

```

### Web méthode des réservations

```

1 public Form1(ServiceDisponibilite.Offre f,String url, String info)
2 {
3
4     InitializeComponent();
5
6     label1.Text = "Id de l'offre : " + f.idOffre;
7     label2.Text = info;
8     pictureBox1.Load(url);
9     textBox1.Text = f.idOffre;
10    label3.Text = "Prix total de l'offre : " + Convert.ToString(f.prixTotalOffre) + "€";
11    prix = Convert.ToInt32(f.prixTotalOffre);
12    label10.Text = "Date de disponibilité : " + "\n" + "Arriver : "
13    + f.deb.ToString("dd/MM/yyyy") + "\n" + "Départ : " + f.fin.ToString("dd/MM/yyyy");
14    ServiceDisponibilite.Hotel infoHotel = Agence1.hotel.getHotel();
15    label11.Text = "Nom de l'hotel : " + infoHotel.nomHotel + "\n" +
16    "Adresse : " + infoHotel.adresseHotel + "\n" + "Etoiles : " + infoHotel.nbEtoiles + "\n"
17    + "Prix nuit/personne : " + infoHotel.prixNuit;
18    label12.Text = "Cette offre vous plaît ? " + "\n" + "Remplissez notre formulaire"
19    + "\n" + "de réservation : ";
20
21 }

```

### Constructeur principal du GUI