



DEPARTEMENT INFORMATIQUE  
DE LA FACULTE DES SCIENCES

Ahmed Kaci et Yanis Allouch

## **Rapport du TP : service web SOAP**



**HMIN210 — Architectures distribuées**

Référent: Abdelhak-Djamel Seriali

**2021**

## Table des matières

|           |   |           |
|-----------|---|-----------|
| <b>1</b>  | <b>Introduction</b>   | <b>2</b>  |
| <b>I</b>  | <b>Mise en situation et première prise en main</b>            | <b>4</b>  |
| 1.1       | Outils pour tester les services web . . . . .                 | 5         |
| 1.2       | Création et consommation d'un service web simple . . . . .    | 8         |
| 1.3       | Utilisation de service web existant . . . . .                 | 9         |
| 1.4       | Conclusion . . . . .  | 13        |
| <b>II</b> | <b>Réalisation du TP</b>                                      | <b>14</b> |
| <b>2</b>  | <b>Création et utilisation de services web</b>                | <b>15</b> |
| 2.1       | Rappel . . . . .  | 15        |
| 2.2       | Diagramme de cas d'utilisation et diagramme de flux . . . . . | 16        |
| 2.3       | Version sans distribution . . . . .                           | 17        |
| 2.4       | Version distribuée . . . . .                                  | 19        |
| 2.5       | Configuration . . . . .                                       | 23        |
|           | <b>Références</b>   | <b>26</b> |

## 1 Introduction

Ce TP est composé de deux parties.

Premièrement, on retrouve les sections 1.1, 1.2 et 1.3 dans le but de se familiariser et d'acquérir les outils de développement C# et ASP.Net pour les Services Web.

Deuxièmement, la section 2 constitue la réalisation de ce TP.

L'objectif est de développer avec ASP.Net et en C#, des Web Services de réservation d'hôtels en ligne. Cette application devra proposer une interface permettant à un utilisateur de saisir des dates et d'autres critères de recherches pour un séjour dans un hôtel.

### Le contexte

Tout d'abord, via le module d'enseignement HMIN210 Architecture Distribuées, nous avons bénéficié d'une formation à la technologie RMI. Par la suite, nous avons bénéficié d'une introduction à la plate-forme .Net, au langage de programmation C# ainsi qu'à la conception de Services Web SOA. Cependant, la documentation n'étant pas suffisante, nous nous sommes armés d'une bibliographie riche et variée.

Nous avons composé la bibliographie de références industrielles et académiques dans les domaines du développement logiciels .Net, C#, des Web Services et API. Par ailleurs, nous avons intégré des références moins populaires, mais qui nous ont aidés.

L'ensemble de la bibliographie est présent à [la fin de ce rapport](#).

Ce TP utilise le langage de programmation C# 9. Nous utilisons les technologies ASP.NET Framework 4.7+ et Visual Studio 2019 version 16.9.1 possédant les outils intégrés « ASP.NET and web development », « .NET desktop development », « Universal Windows Platform development » et « .NET Core cross-platform development » sous Windows 10 version 2004 qui peuvent expliquer des différences visuelles sur nos captures d'écrans.

Nous avons trouvé une bonne généalogie de la plate-forme .Net, C# et framework sous-jacent expliqué dans l'excellente ressource pédagogique [Pri20] dans le chapitre 1.

Notre choix est motivé par le besoin de compatibilité par la contrainte d'utilisation de .Net Framework par la suite sur le choix des technologies des web services.

Nous avons personnellement apprécié la lecture de [Wee+05] nous donnant la connaissance et la compréhension par les artisans des WS-\* et des concepts clés liés, intrinsèquement comme BPEL les chorégraphies, l'orchestration, **en revanche, elles n'ont pas été mises en place.**

Sans compter sur le bienfait des lectures [Hig15], [Mas11], [Sto15] et [WPR10] excepté que leur effort est tourné autour des services REST que nous réutiliserons pour le TP suivant, évidemment sur REST.

Nous n'aurions pas compris les différences entre SOA et Microservices dans leur objectif et leur conception autrement que par le recul fourni dans un rapport comparant les deux [Ric16] recommandé par l'entreprise [Nginx](#).

En revanche, les contraintes de temps et la portée de ce TP ne nous ont pas permis d'étudier les designs patterns pour les SOA expliqués dont le livre [Er108].

## Première partie

### Mise en situation et première prise en main

## 1.1 Outils pour tester les services web

L'exercice 1, nous présente deux outils : SoapUI et Postman.

SoapUI est une application open-source qui permet de tester des web services dans une architecture orientée services (SOA).

Les fonctionnalités de cette application incluent « l'inspection des web service, l'invocation, le développement, la simulation, le mocking, les tests fonctionnels, les tests de montée en charge et de conformité »<sup>1</sup>.

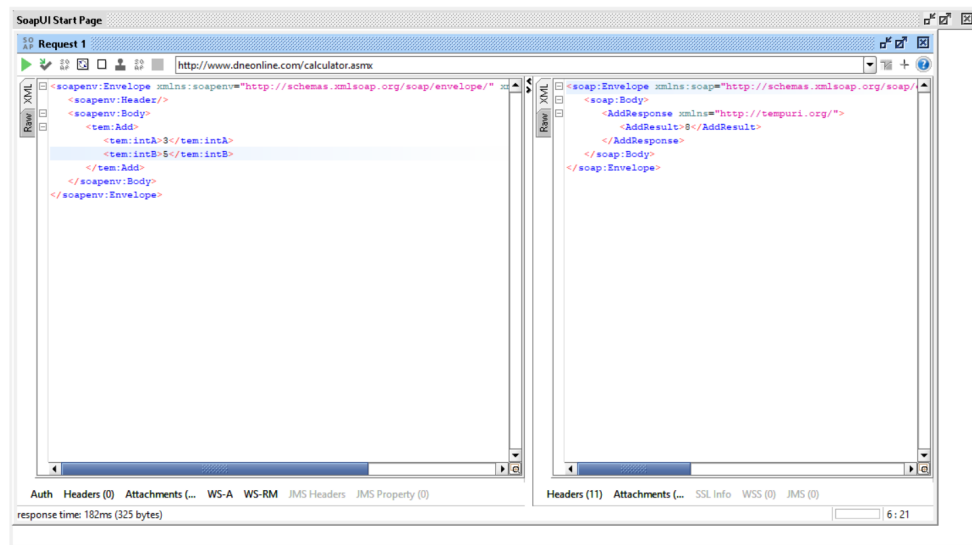


FIGURE 1 – Capture d'écran du résultat du tutoriel "SoapUI : Getting Started"

1. <https://fr.wikipedia.org/wiki/SoapUI>

Postman est une application ayant été auparavant distribué comme une extension du navigateur Google Chrome.

Il est aujourd'hui distribué et commercialisé comme une version standalone. Il permet l'automatisation de test d'API ainsi que le développement et la validation des API par l'équipe de développement.

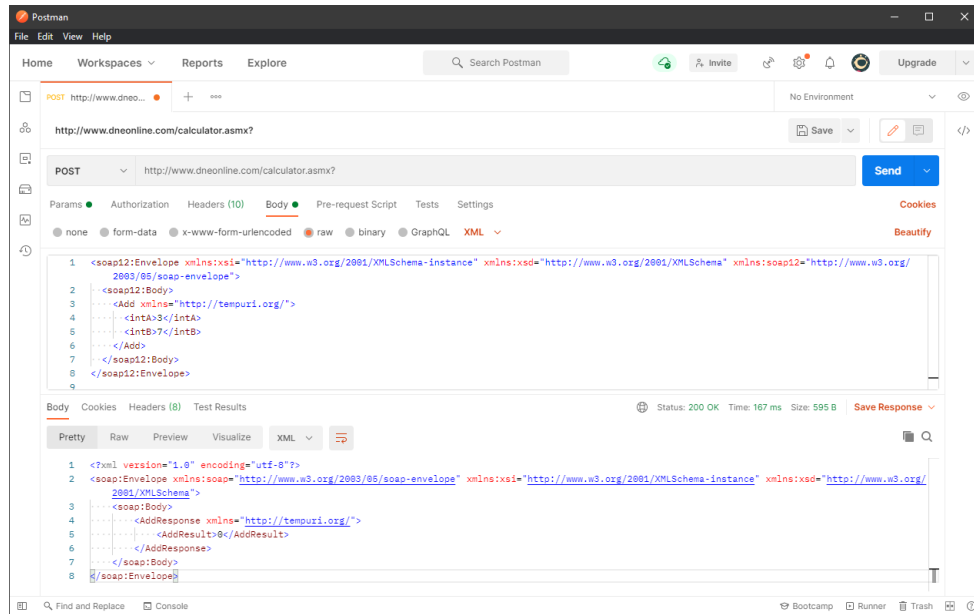


FIGURE 2 – Capture d'écran du résultat du tutoriel Postman : "Making SOAP requests"

## Conclusion

Selon les messages d'utilisateurs professionnels avec plusieurs années d'expérience sur [Quora](#) et l'article fournis sur [Katalon](#).

Nous pouvons conclure que SoapUI et Postman sont les deux outils les plus cités de test d'API. Ils sont utilisés pour tester et gérer des APIs. Ceux-ci aident les développeurs à créer et partager des rapports pendant le développement. Bien que Postman et SoapUI soient activement utilisés par de nombreuses entreprises, ceux-ci présentent plusieurs avantages et inconvénients :

- Les deux outils sont populaires, mais dans de nombreux cas **SoapUI** est préféré par **les grandes entreprises** et **Postman** est préféré par les **entreprises de taille moyenne** ;
- Cela peut être dû au facteur **d'accessibilité**. Postman est disponible à un prix décent 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 fixé à **\$659 par année** ;
- Postman est facile pour partager des rapports et les données des APIs et fournit des outils de **collaborations solides** ;
- Alors que SoapUI est **une interface simple** qui permet de faire du glisser-déposer, de pointer et de cliquer sur l'option à configurer. Il est également bon dans l'intégration d'outils, offre un moyen facile de **réutiliser les scripts de test**. Cet outil est la meilleure option lorsqu'il y a moins de professionnels impliqués dans la gestion des API ;
- Ces deux outils populaires ont également des limites par exemple Postman possède une **interface utilisateur complexe** et une **documentation médiocre**. De même pour SoapUI Pro, qui possède des **problèmes de performances**, de ralentissement dû à la gestion d'énormes données et la consommation d'une grande quantité de mémoire ;
- Contrairement à SoapUI, Postman est considéré comme plus utile pour les API RESTful principalement lorsque nous répétons des requêtes, tout en effectuant des **tests automatisés**, etc.



## 1.2 Création et consommation d'un service web simple

L'exercice 2 nous guide pour mettre en place notre premier Web Service.

Ce qui suit, correspond aux captures d'écran des différentes étapes de la réalisation du second exercice de ce TP qui consiste à suivre le tutoriel fourni sur la [documentation Microsoft™](#).

La première étape a pour but de réaliser un service MathService basique (Figure 3) qui en se basant sur l'échange de requêtes SOAP, permettent de faire les quatre opérations de calcul de bases suivantes : l'addition, la soustraction, la multiplication et la division.

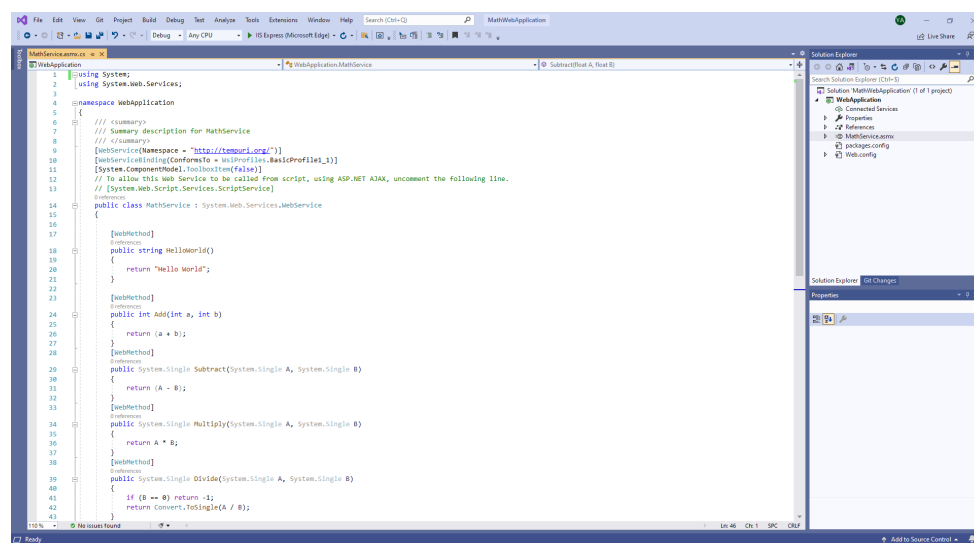


FIGURE 3 – Capture d'écran du code du Web service MathService.asmx

La seconde partie consiste à créer une application de type console qui va consommer le web service précédemment développé.

Le tutoriel nous guide dans le développement de l'application (Figure 5) la plus simple pour consommer notre web service.

L'exécution de notre application permet d'échanger une requête SOAP avec les paramètres **intA** équivalant à 2 et **intB** équivalant à 4.

Notre web service nous rend donc le résultat (Figure 6) **intA + intB** équivalant à 6.

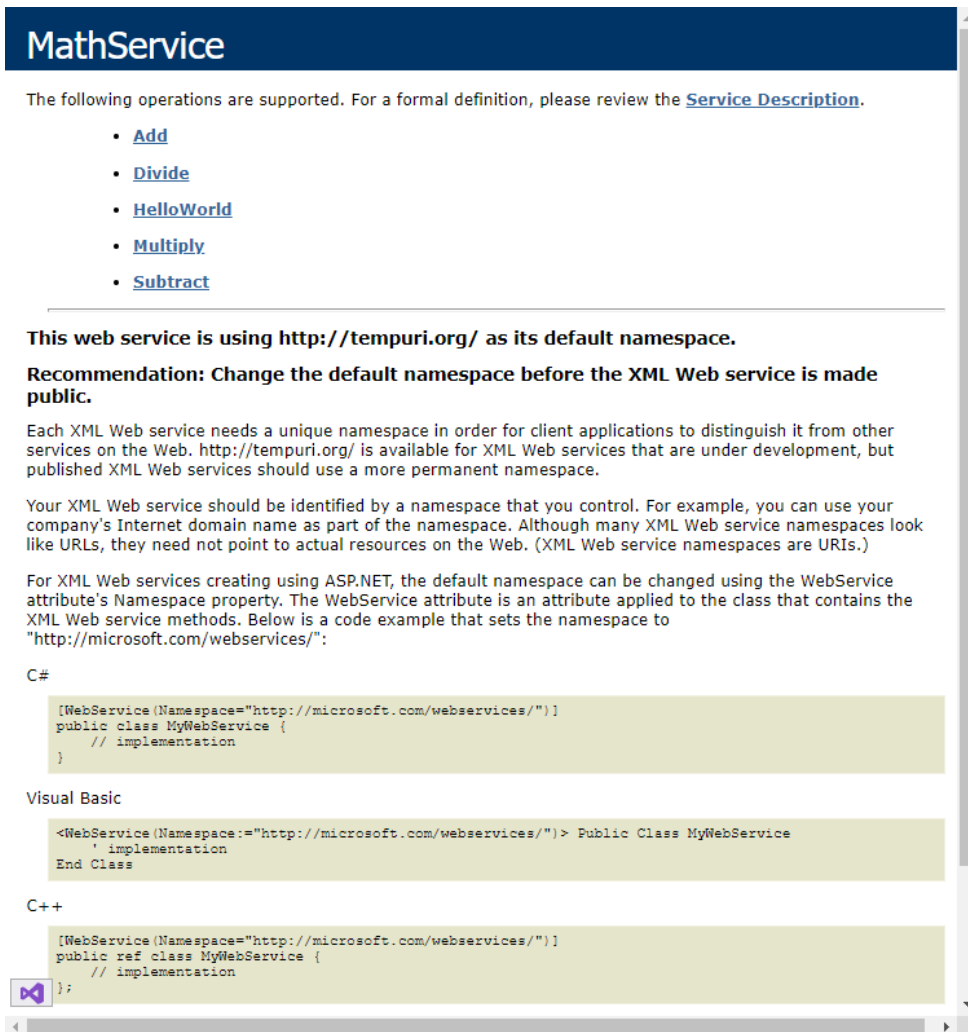


FIGURE 4 – Capture d’écran de la page d’aide du Web service MathService.asmx

### 1.3 Utilisation de service web existant

L’exercice 3 nous présente plusieurs Web Service à consommer.

La WSDL du web service [calculatrice](#) fournit le mécanisme à travers lequel nous retrouvons la définition des méthodes exposées au public afin de pouvoir implémenter une application conforme à la spécification fournie.

WSDL décrit les types de données et les structures utilisées par le web service. Il présente comment les types et les structures sont liés et échangés en spécifiant l’opération concernée.

La WSDL du web service [calculatrice](#) possède les opérations d’addition (**Add**), de soustraction (**Subtract**), de division (**Divide**) et de multiplication (**Multiply**).

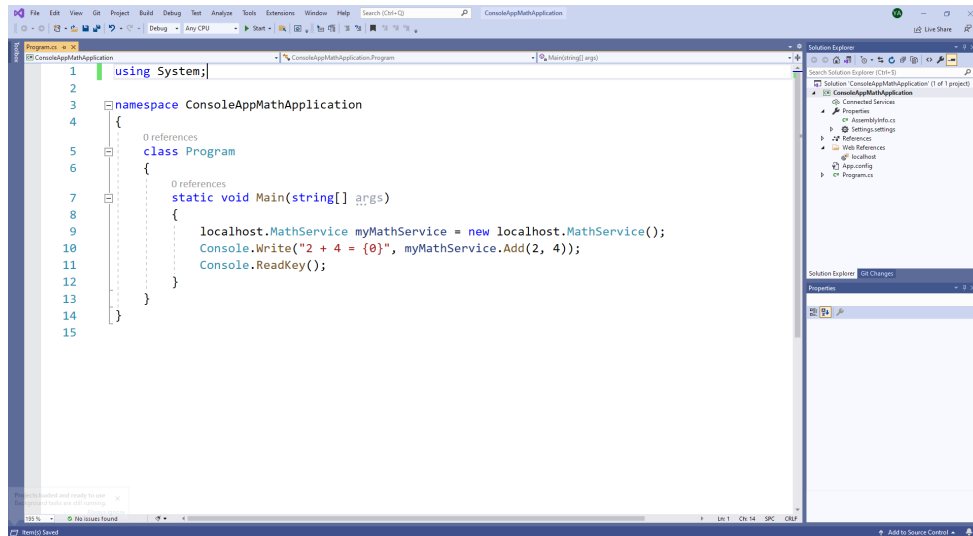


FIGURE 5 – Capture d’écran du code du consommateur du Web service MathService

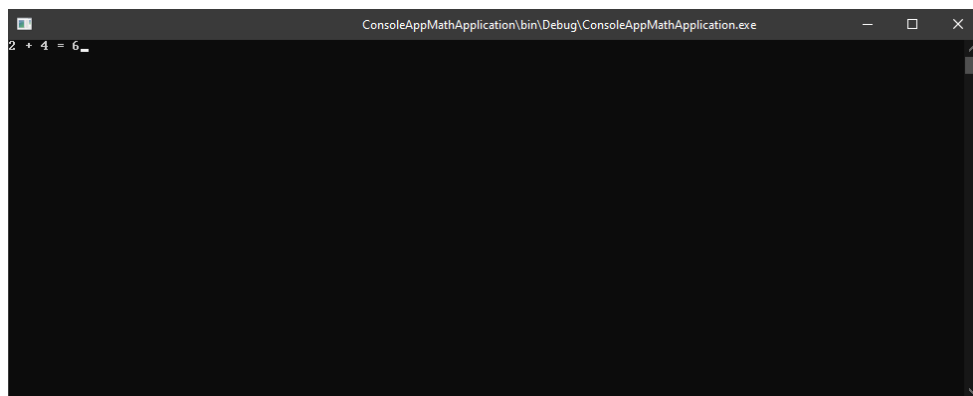


FIGURE 6 – Capture d’écran du résultat du consommateur du Web service MathService

La capture d’écran de la WSDL du service est visible sur la figure 7.

Nous proposons alors une application de type console C# qui utilise et implémente les opérations du web service [calculatrice](#), visible en figure 8.

La capture d’écran d’exécution de l’application est visible sur la figure 9.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<wsdl:definitions targetNamespace="http://tempuri.org/">
  <wsdl:types/>
  <wsdl:message name="AddSoapIn"/>
  <wsdl:message name="AddSoapOut"/>
  <wsdl:message name="SubtractSoapIn"/>
  <wsdl:message name="SubtractSoapOut"/>
  <wsdl:message name="MultiplySoapIn"/>
  <wsdl:message name="MultiplySoapOut"/>
  <wsdl:message name="DivideSoapIn"/>
  <wsdl:message name="DivideSoapOut"/>
  <wsdl:portType name="CalculatorSoap"/>
  <wsdl:binding name="CalculatorSoap" type="tns:CalculatorSoap"/>
  <wsdl:binding name="CalculatorSoap12" type="tns:CalculatorSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Add">
      <soap12:operation soapAction="http://tempuri.org/Add" style="document"/>
      <wsdl:input/>
      <wsdl:output/>
    </wsdl:operation>
    <wsdl:operation name="Subtract">
      <soap12:operation soapAction="http://tempuri.org/Subtract" style="document"/>
      <wsdl:input/>
      <wsdl:output/>
    </wsdl:operation>
    <wsdl:operation name="Multiply">
      <soap12:operation soapAction="http://tempuri.org/Multiply" style="document"/>
      <wsdl:input/>
      <wsdl:output/>
    </wsdl:operation>
    <wsdl:operation name="Divide">
      <soap12:operation soapAction="http://tempuri.org/Divide" style="document"/>
      <wsdl:input/>
      <wsdl:output/>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Calculator"/>
</wsdl:definitions>

```

FIGURE 7 – Capture d'écran de la WSDL du Web service Calculatrice

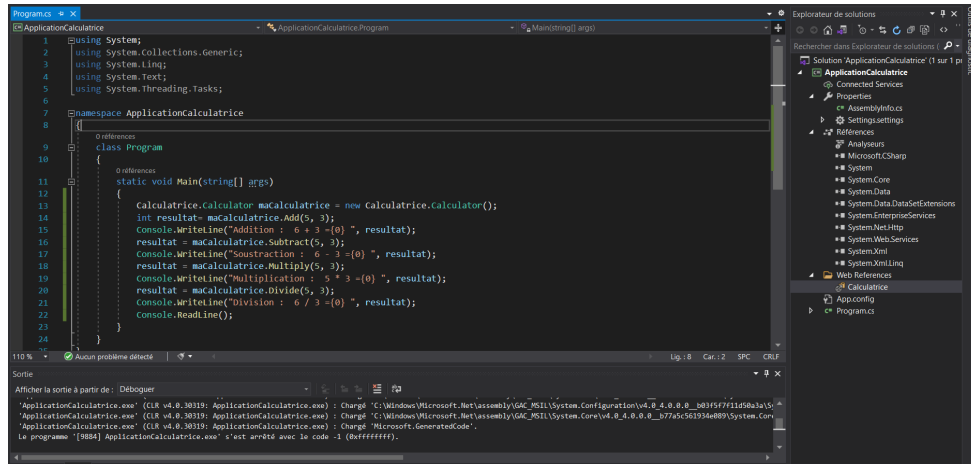


FIGURE 8 – Capture d’écran du code de l’application qui implémente les opérations du web service

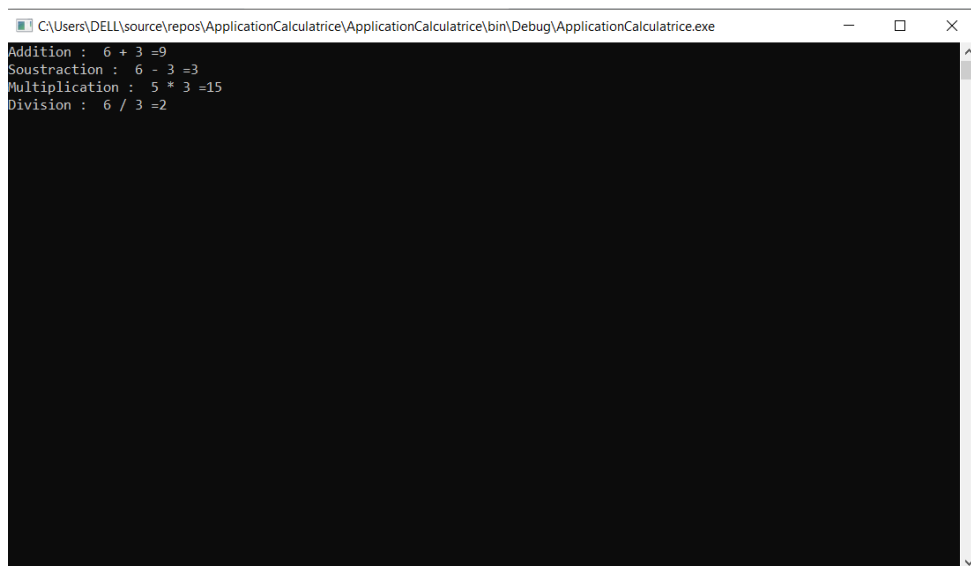


FIGURE 9 – Capture d’écran de l’exécution de l’application

## **1.4 Conclusion**

En somme, cette première partie nous a permis de comprendre comment mettre en place un Web Service, le consommer via une application console C# et de pouvoir tester le Web service mise en place, autrement que par notre application console en utilisant SoapUI ou Postman.

Deuxième partie

Réalisation du TP

## 2 Création et utilisation de services web

### Introduction

Étant donnée, que deux versions sont attendues, nous avons décidé de séparer le projet en deux dépôts Git hébergés sur <https://www.gitlab.com> :

1. <https://gitlab.com/yanisallouch/gestionhotelcentraliser> (sans web service) ;
2. <https://gitlab.com/kaciahmed3/gestionhoteldistribueparagencedevoyage> (avec WS Soap).

Les adresses mail de notre référent M. Abdelhak-Djamel Seriai fourni aux pré-alables ont été ajoutées aux dépôts susmentionnés :

- rechercheseriai@gmail.com ;
- seriai@lirmm.fr ;
- abdelhak.seriai@umontpellier.fr.

### 2.1 Rappel

L'objectif est de développer en C# une application de réservation d'hôtels en ligne. 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 ces 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'étoiles, nombre de lits proposés.

L'utilisateur choisira un hôtel dans la liste proposée et l'application lui demandera les informations suivantes : le nom et le 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.



## 2.2 Diagramme de cas d'utilisation et diagramme de flux

Nous avons schématisé notre compréhension de la consigne ci-dessus par un diagramme de cas d'utilisation (Voir figure 10) et un diagramme de flux (Voir la figure 11). Cela nous a permis de délimiter les fonctionnalités de la version centralisée de notre système, ainsi que le schéma global de sa réalisation.

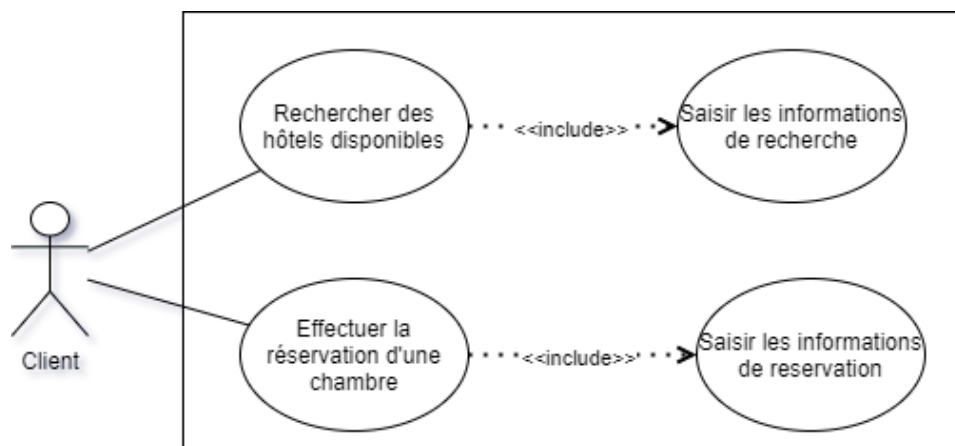


FIGURE 10 – Diagramme UML use case de l'application

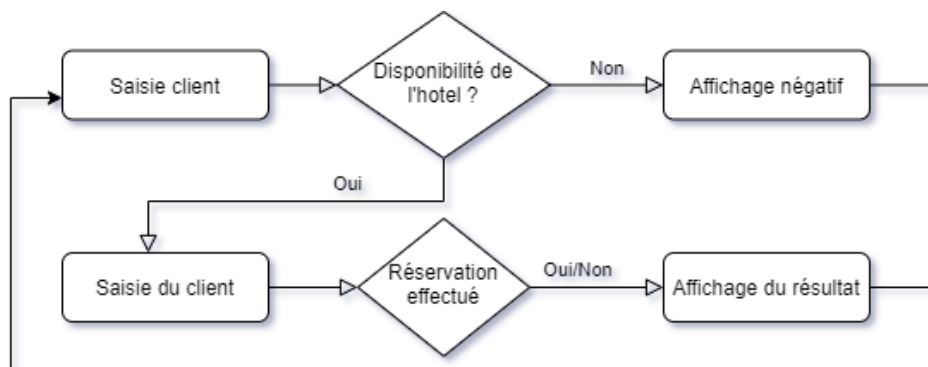


FIGURE 11 – Diagramme de flux de l'application

## 2.3 Version sans distribution

Le premier [dépôt](#) répond à la première question.

### Définition et présentation de l'architecture mise en place

Grâce aux informations précédentes, nous savons ce que nous allons modéliser grâce aux structures OO<sup>2</sup> de C#.

Voici le diagramme de classes en figure 12.

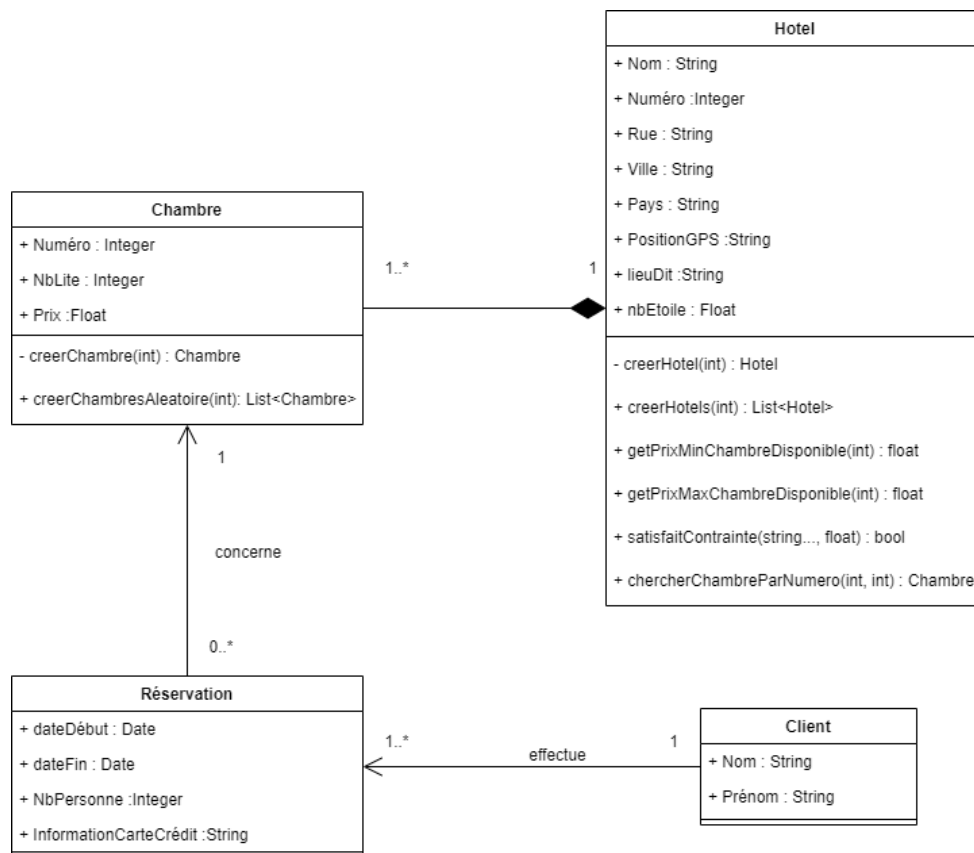


FIGURE 12 – Diagramme de classes de l'application centralisée

### Description du diagramme de classe

Tout naturellement, une chambre est caractérisée par un numéro de porte dans son hôtel.

D'une part nous savons qu'une chambre est composée d'un certain nombre de lits.

---

2. Object Oriented

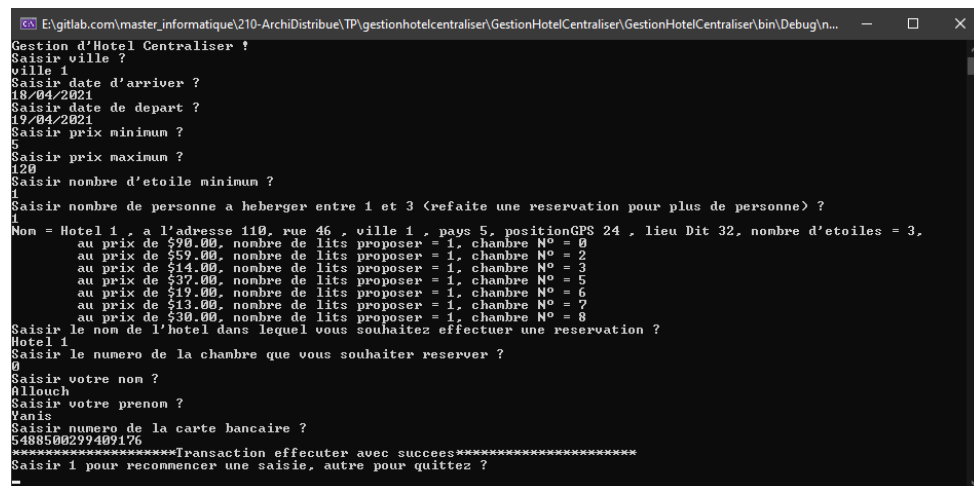
Selon le sujet nous aurions dû créer une classe ou une énumération *TypeChambre* caractérisant les différentes quantités, cependant nous n'avons pas trouvé d'intérêt dans cette abstraction et l'avons omise.

D'autre part, une chambre possède un prix lui étant propre, nous avons pensé qu'avec les mêmes caractéristiques, une chambre bien orientée face piscine était valorisée par rapport a une chambre mal orientée dans le même hôtel.

La classe Réservation a été pensée pour pouvoir sauvegarder l'historique des réservations de chaque chambre. Et ce, afin de pouvoir les fournir à la demande des hôtels. Même si cela ne figure pas dans l'énoncé, nous avons jugé que cela pourrait être utile. Une réservation possède, donc, toutes les informations nécessaires, telles que le nombre de personnes a hébergé, ainsi que la date de début et de fin de la réservation.

De surcroît elle concerne un client, une chambre et par transitivité un hôtel.

## Exécution



```

E:\gitlab.com\master_informatique\210-ArchiDistribue\TP\gestionhotelcentraliser\GestionHotelCentraliser\GestionHotelCentraliser\bin\Debug\n...
Gestion d'Hotel Centraliser !
Saisir ville ?
ville 1
Saisir date d'arriver ?
18/04/2021
Saisir date de depart ?
19/04/2021
Saisir prix minimum ?
5
Saisir prix maximum ?
120
Saisir nombre d'etoile minimum ?
1
Saisir nombre de personne a heberger entre 1 et 3 (refaite une reservation pour plus de personne) ?
1
Non = Hotel 1 , a l'adresse 110, rue 46 , ville 1 , pays 5, positionGPS 24 , lieu Dit 32, nombre d'etoiles = 3.
    au prix de $90.00, nombre de lits proposer = 1, chambre N° = 0
    au prix de $59.00, nombre de lits proposer = 1, chambre N° = 2
    au prix de $14.00, nombre de lits proposer = 1, chambre N° = 3
    au prix de $37.00, nombre de lits proposer = 1, chambre N° = 5
    au prix de $19.00, nombre de lits proposer = 1, chambre N° = 6
    au prix de $13.00, nombre de lits proposer = 1, chambre N° = 7
    au prix de $30.00, nombre de lits proposer = 1, chambre N° = 8
Saisir le nom de l'hotel dans lequel vous souhaitez effectuer une reservation ?
Hotel 1
Saisir le numero de la chambre que vous souhaitez reserver ?
0
Saisir votre nom ?
Allouch
Saisir votre prenom ?
Yanis
Saisir numero de la carte bancaire ?
5488500299409176
*****Transaction effecuter avec succes*****
Saisir 1 pour recommencer une saisie, autre pour quittez ?
-

```

FIGURE 13 – Capture d'écran du processus de recherche et de réservation

## 2.4 Version distribuée

Le second [dépôt](#) réponds a la seconde question.

### Mise a jour du diagramme UML

Voici le diagramme de classe de l'application mis à jour de sorte à intégrer le fait qu'un hôtel peut avoir des agences partenaires qui chacune bénéficient d'un tarif propre.

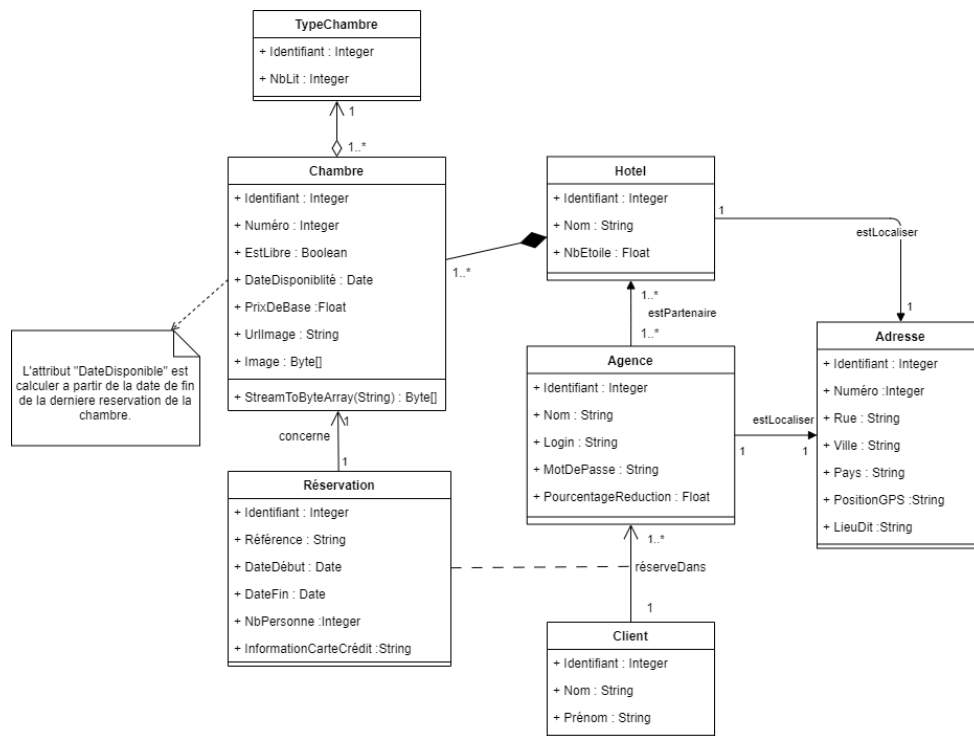


FIGURE 14 – Diagramme de classe de l'application avec agence partenaire

## Explication du travail réalisé

Afin de réaliser la version distribuée de l'application, cette dernière a été découpée en trois parties.

La première partie est l'application cliente (fournit à l'agence) qui communique avec deux WS<sup>3</sup> selon son besoin :

- En outre, elle communique avec le premier WS pour obtenir la liste des chambres disponibles.
- en plus, elle communique avec le second WS pour effectuer une réservation.

Dans la seconde partie de l'application, nous avons implémenté les deux WS introduits précédemment. Ces WS, à savoir le WS qui permet de récupérer les informations sur la disponibilité des chambres des hôtels et le WS qui permet d'effectuer une réservation reçue par les applications clientes, communiquent avec un troisième WS utilisé pour la gestion des données.

La troisième partie de l'application est composée d'un WS qui pourra être relié à un SGBD afin de gérer de façon persistante les données utilisées pour assurer le bon fonctionnement du système.

Le schéma suivant (figure 15) permet d'illustrer le travail réalisé.

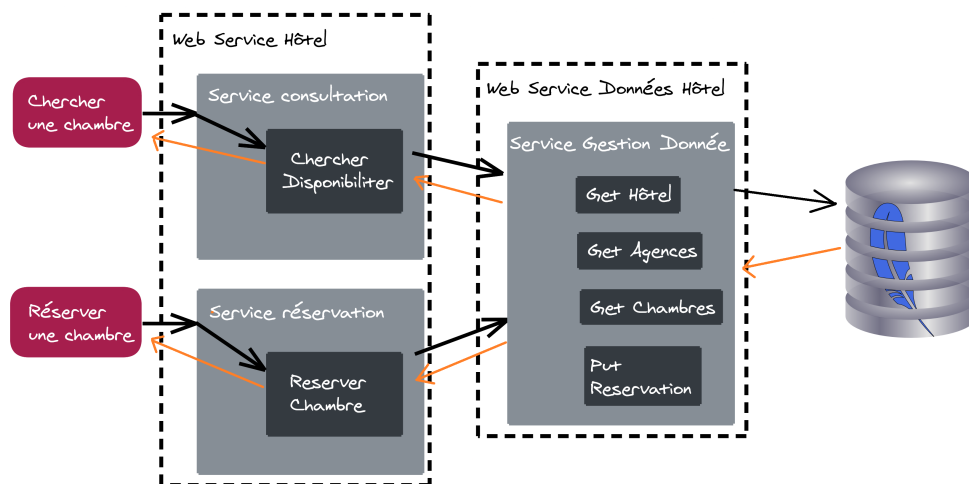


FIGURE 15 – Schéma de l'architecture du système

3. Abréviation pour Web Service.

## Le service web intègre des images

Pour répondre à cette problématique, nous avons cherché un moyen de convertir une image pour l'inclure dans une offre.

Pour ce faire, nous faisons une conversion d'une image vers un *byte[]* en utilisant les fonctions de la librairie **System.Drawing.dll** qu'il faut alors inclure dans le projet.

La conversion consiste à ouvrir le fichier de l'image dans une instance de la classe Image dans la fonction **StreamToByteArray** qu'on a décrit dans la classe **Chambre** et de construire octet par octet la conversion.

Quelques exécution de notre application et nos fichiers nous ont permis de déduire qu'un buffer de taille 64\*1024 octet pour faire les écritures et lectures du fichier était rapide, autrement en dessous de 1024 octets l'application prenait plusieurs minutes pour traiter une image. Cette optimisation est nécessaire parce qu'il n'est pas tolérable d'attendre 10 minutes pour traiter 3 chambres par requête !

Le chemin des images étant codé en dur, nous expliquons dans la section 2.5 : Configuration, à quels endroits du code il faudrait le paramétrer pour pouvoir ré-exécuter le projet dans n'importe quel environnement.

## Base de données et interface graphique

Concernant la base de données nous avons utilisé le SGBD SQLite version 3. Il nous a fallu importer via le gestionnaire de paquet Nuget pour Solutions : **SQLite**.

Voici les liens vers les tutoriels qu'on a utilisé pour pouvoir configurer SQLite de notre côté :

- [C# Database Connection: How to connect SQL Server \(Example\)](#)
- [Getting started with SQLite in C#](#)
- [Connect C# Application to SQLite Database and interact with it](#)
- [How to execute an .SQL script file using C#](#)
- [How to use Console.WriteLine in ASP.NET \(C#\) during debug?](#)
- [Create SQLite Database and table](#)
- [Reading int values from SqlDataReader](#)

En outre pour l'interface graphique, nous avons utilisé la technologie Windows Forms.

Nous l'avons choisi pour ses qualités de prototypage rapide et puissant et puis surtout pour la facilité de mise en place pour nos besoins et contrainte de temps.

De nouveau, voici les ressources utilisées :

- [C# Windows Forms Application Tutorial with Example](#)
- [Change how a date or time is displayed in a date picker, text box, ...](#)
- [How to set the Format of the DateTimePicker in C#?](#)
- [Set default format of datetimepicker as dd-MM-yyyy](#)
- [How to i remove the border line of a Group Box?](#)
- [Max Char in TextBox C#?](#)
- [how to fill date picker with dates come from Database?](#)
- [C# DateTimePicker Control](#)
- [How do I Pass data between TabControl, Tabpages and UserControl](#)
- [How to switch tabs with button in a TabControl?](#)

- [Working with Windows TabControl with C#](#)
- [C# datetimepicker : how do i get the date in the format? 1/1/2001](#)

Nous avons ajouté une feature permettant de visualiser l'image en plein écran, si vous le souhaitez!

Pour ce faire, nous sommes basées sur les explications d'utilisateur de StackOverflow et CSharpCorner suivant :

- [How to Open a Second Form Using First Form in Windows Forms](#)
- [How do I change a PictureBox's image?](#)
- [Using PictureBox In Windows Forms](#)
- [Fit Image into PictureBox](#)

Son fonctionnement est simple, il suffit de cliquer sur la miniature affiché par le logiciel. Une forms s'ouvre alors en plein écran, cliquer de nouveaux dessus pour la fermer.

## Captures d'écran de l'exécution de l'application

### Version sans les interfaces graphiques et sans SGBD

```

E:\gitlab.com\gestionhoteldistribueeparagencedevoyage\Agence1\Agence1\bin\Debug\Agence1.exe
Saisir date d'arriver ?
18/06/2021
Saisir date de depart ?
19/06/2021
Saisir nombre de personne a heberger entre 1 et 3 (refaite une reservation pour plus de personne) ?
1
identifiant Offre : 1_1, Date Disponibilit  : 01/03/2021, Type Chambre (nombre de lits) :1, Prix :46.4099998176098
identifiant Offre : 1_15, Date Disponibilit  : 10/03/2021, Type Chambre (nombre de lits) :1, Prix :37.3099998533726
identifiant Offre : 1_12, Date Disponibilit  : 10/04/2021, Type Chambre (nombre de lits) :1, Prix :62.7899997532368
Saisir voulez vous continuer (1 = oui / -1 = non) ?
1

```

FIGURE 16 – Capture d'écran de la console Agence lors d'une recherche

```

E:\gitlab.com\gestionhoteldistribueeparagencedevoyage\Agence1\Agence1\bin\Debug\Agence1.exe
Saisir date d'arriver ?
18/06/2021
Saisir date de depart ?
19/06/2021
Saisir nombre de personne a heberger entre 1 et 3 (refaite une reservation pour plus de personne) ?
1
identifiant Offre : 1_1, Date Disponibilit  : 01/03/2021, Type Chambre (nombre de lits) :1, Prix :46.4099998176098
identifiant Offre : 1_15, Date Disponibilit  : 10/03/2021, Type Chambre (nombre de lits) :1, Prix :37.3099998533726
identifiant Offre : 1_12, Date Disponibilit  : 10/04/2021, Type Chambre (nombre de lits) :1, Prix :62.7899997532368
Saisir voulez vous continuer (1 = oui / -1 = non) ?
1
Saisir votre nom ?
 llouch
Saisir votre pr nom ?
Ahmed
Saisir les informations de votre carte de cr dit ?
4999999999999999
Veuillez s lectionner l'identifiant de l'offre que vous souhaitez, SUP (pour quitter tapez -1)
Saisir identifiant de l'offre ?
1_1
R servation r uss .
La r f rence de votre r servation est :#4221
Saisir voulez effectuer une autre r servation (1 = oui / -1 = non) ?
-1

```

FIGURE 17 – Capture d'écran de la console Agence lors d'une r servation

```

E:\gitlab.com\gestionhotel\distribueparagencedevoyage\Agence\bin\Debug\Agence1.exe
Saisir date d'arriver ?
18/06/2021
Saisir date de depart ?
19/06/2021
Saisir nombre de personne a heberger entre 1 et 3 <refaite une reservation pour plus de personne> ?
1
identifiant Offre : 1_1, Date Disponibilit  : 01/03/2021, Type Chambre <nombre de lits> :1, Prix :46.4099998176098
identifiant Offre : 1_15, Date Disponibilit  : 10/03/2021, Type Chambre <nombre de lits> :1, Prix :37.3099998533726
identifiant Offre : 1_12, Date Disponibilit  : 10/04/2021, Type Chambre <nombre de lits> :1, Prix :62.7899997532368
Saisir voulez vous continuer <1 = oui / -1 = non> ?
1
Saisir votre nom ?
Allouch
Saisir votre pr nom ?
Ahmed
Saisir les informations de votre carte de cr dit ?
4999999999999999
Veuillez s lectionner l'identifiant de l'offre que vous souhaitez, SUP <pour quitter tapez -1>
Saisir identifiant de l'offre ?
1_1
R servation r ussie.
La r f rence de votre r servation est :#7942
Saisir voulez effectuer une autre r servation <1 = oui / -1 = non> ?
-1
Merci de votre visite ! Appuyez sur une touche pour quitter

```

FIGURE 18 – Capture d cran de la console Agence de fin d'ex cution

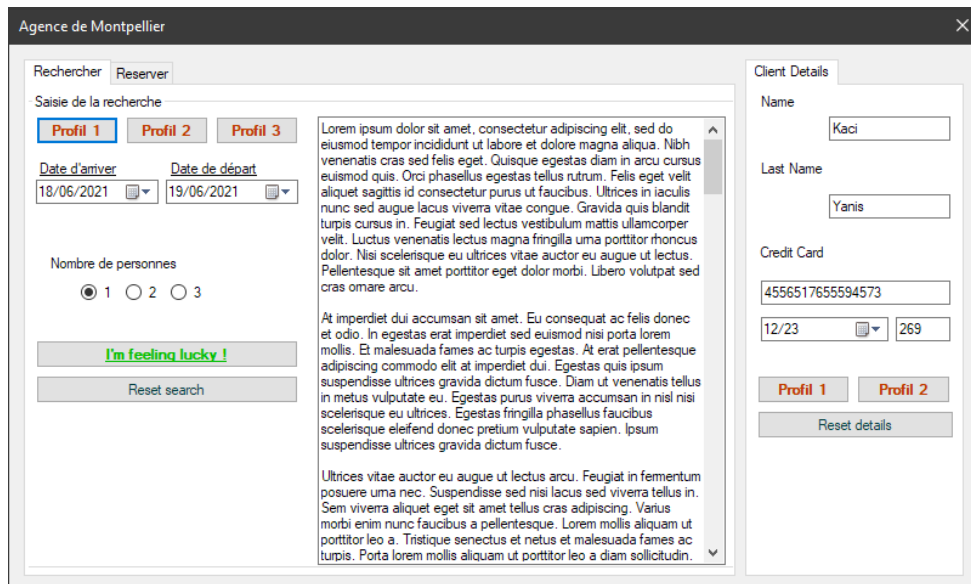


FIGURE 19 – Capture d cran de l'application Agence a l'ouverture

## Version avec les interfaces graphiques et avec SGBD

### 2.5 Configuration

Le dossier assets contient les images des chambres, le fichier SQL qui permet de cr er les tables de la base de donn es et le fichier SQL qui permet d'ins rer les donn es de tests dans la BDD.

Nous d crivons succinctement les diff rentes configurations par release :

#### 1. Version avec images

- L'introduction des images par le web service, contraint l'h bergeur du WS a modifier les chemin vers les images qui est  crit en dur dans le fichier. `/ProjetGestionDonnee/Database.cs`. Il faudra modifier la variable **pathDossierAChanger** qui devra mener jusqu'  la racine du projet `/gestionhotel-distribueparagencedevoyage`. Inspirez-vous du chemin d j  utilis .



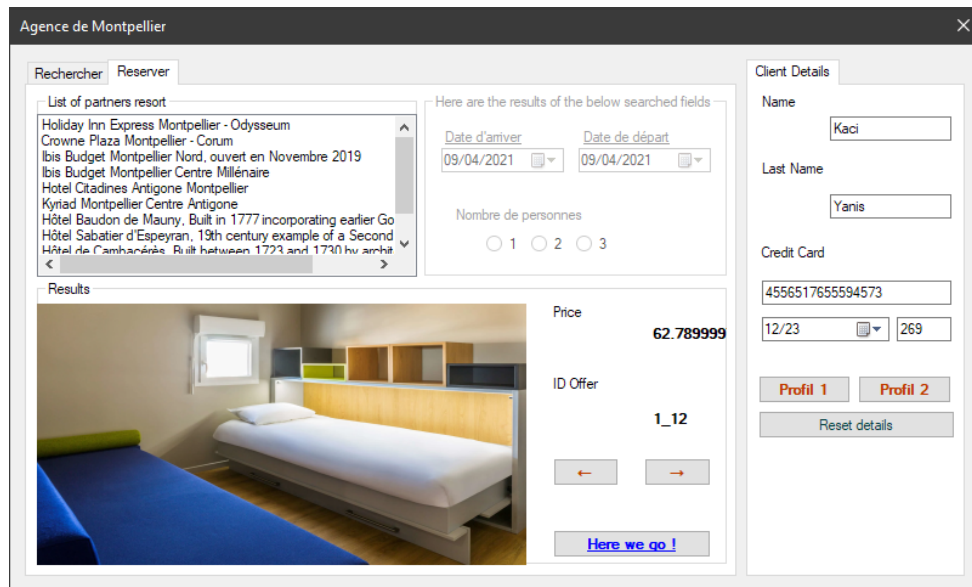


FIGURE 20 – Capture d'écran de l'application Agence après une recherche

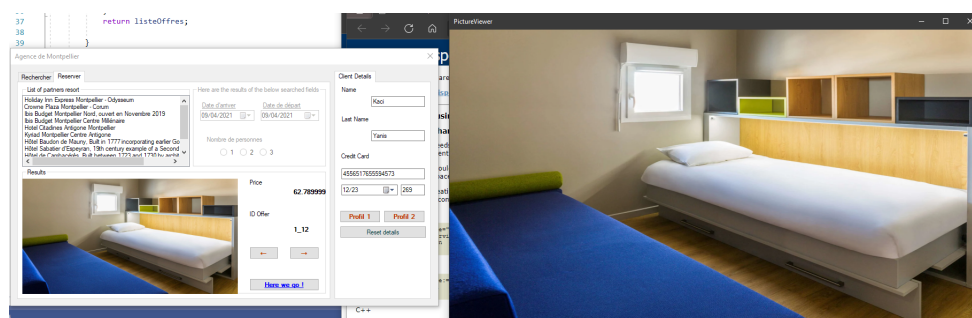


FIGURE 21 – Capture d'écran de l'application Agence avec visualisation en plein écran d'une image

## 2. Version avec images dans un SGBD

- L'introduction d'un SGBD par le web service, délocalise l'écriture en dur du chemin vers les images dans la base de données. Il faudra modifier les insertions SQL de la table `Chambre`, contenu dans le fichier. `/gestionhoteldistribueparagencedevoyage/assets/insert.sql`.

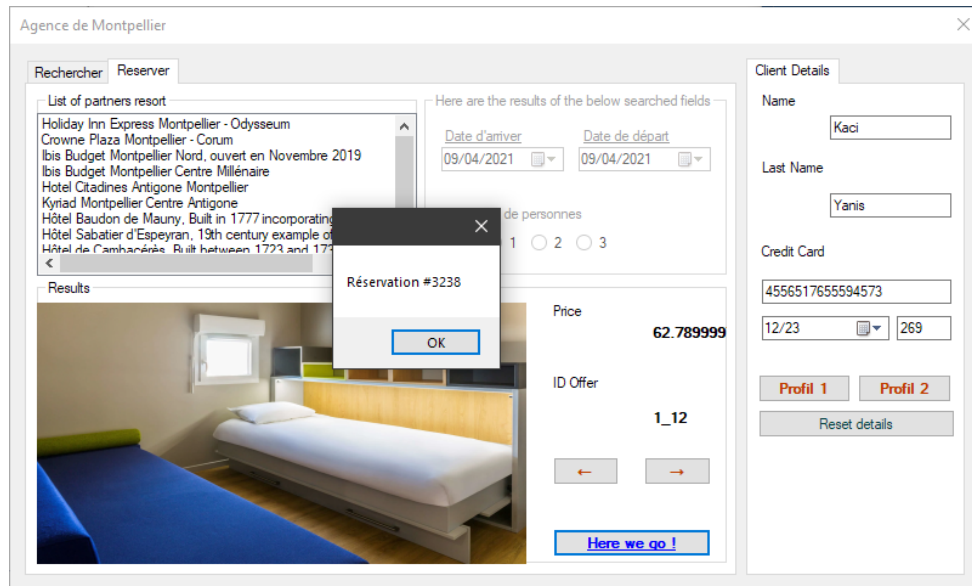


FIGURE 22 – Capture d'écran de l'application Agence avec réservation

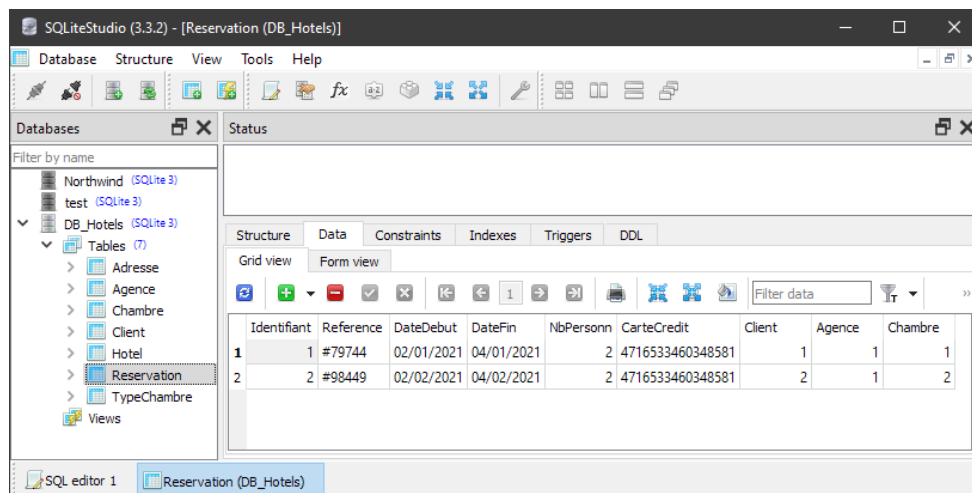


FIGURE 23 – Capture d'écran des données de la table Réservation vu par SQLite Studio

## Références

- [EG05] ERIC, N. et GREG, L. *understanding SOA with web services*. Addison Wesley Professional, 2005.
- [Erl08] ERL, T. *SOA Design Patterns (paperback)*. Pearson Education, 2008.
- [Hig15] HIGGINBOTHAM, J. *Designing Great Web APIs*. O'Reilly Media, Incorporated, 2015.
- [Hur+09] HURWITZ, J. S. et al. *Service Oriented Architecture (SOA) for Dummies*. John Wiley & Sons, 2009.
- [JBW12] JACOBSON, D., BRAIL, G. et WOODS, D. *APIs : A strategy guide*. " O'Reilly Media, Inc.", 2012.
- [KW14] KURTZ, J. et WORTMAN, B. *ASP. NET Web API 2 : Building a REST Service from Start to Finish*. Apress, 2014.
- [Mas11] MASSE, M. *REST API Design Rulebook : Designing Consistent RESTful Web Service Interfaces*. " O'Reilly Media, Inc.", 2011.
- [Pri20] PRICE, M. J. *C# 9 and .NET 5 – Modern Cross-Platform Development - Fifth Edition*. packt, 2020.
- [Ric+13] RICHARDSON, L. et al. *RESTful Web APIs : Services for a Changing World*. " O'Reilly Media, Inc.", 2013.
- [Ric16] RICHARDS, M. *Microservices vs. Service-Oriented Architecture*. O'Reilly Media, Incorporated, 2016.
- [SR19] SUBRAMANIAN, H. et RAJ, P. *Hands-On RESTful API Design Patterns and Best Practices : Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs*. Packt Publishing Ltd, 2019.
- [Sto15] STOWE, M. *Undisturbed REST : A guide to designing the perfect API*. MuleSoft, 2015.
- [Wee+05] WEERAWARANA, S. et al. *Web services platform architecture : SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Prentice Hall PTR, 2005.
- [WPR10] WEBBER, J., PARASTATIDIS, S. et ROBINSON, I. *REST in practice : Hypermedia and systems architecture*. " O'Reilly Media, Inc.", 2010.