



DEPARTEMENT INFORMATIQUE
DE LA FACULTE DES SCIENCES

Ahmed Kaci et Yanis Allouch

Rapport du TP : service web REST



HMIN210 — Architectures distribuées

Référent: Abdelhak-Djamel Seriali

2021

Table des matières

1	Introduction	2
I	Mise en situation et première prise en main	4
1.1	Tester des Services Web REST	5
II	Réalisation du TP	8
2	Création et utilisation de services web	9
2.1	Rappel	9
2.2	Diagramme des cas d'utilisations et diagramme de flux	10
2.3	Version distribuée-Agence de voyage et hôtels	11
2.4	Le service web intègre des images	14
2.5	Un service web proposé par les agences de voyage pour comparateurs . . .	16
2.6	Base de données	19
III	Conclusion	22
2.7	Tester des services REST	23
2.8	Création et consommation de service REST	23
2.9	Configuration	25
	Références	26

1 Introduction

Ce TP est composé de deux parties.

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

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

L'objectif est de développer avec Web Api 2 et ASP.Net en C#, des Web Services de comparaison et 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 via un comparateur d'agence de réservation.

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] dont les efforts sont tournés autour des services REST que nous appliquons pour ce TP 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] appuyé par l'entreprise [Nginx](#) aux éditions O'Reilly.

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 dans le livre [\[Er108\]](#).

Première partie

Mise en situation et première prise en main

1.1 Tester des Services Web REST

L'exercice 1, nous présente plusieurs API à tester avec l'outil [Postman](#) introduit dans le TP, précédent, qui nous permet de maîtriser la technologie SOAP :

- Quelques service REST sur l'annuaire RAPIDAPI : <https://rapidapi.com/>;
- Instagram :
 - <https://developers.facebook.com/docs/instagram-basic-display-api>;
 - <https://developers.facebook.com/docs/instagram-api/-Gmail>;
- Gmail :
 - <https://developers.google.com/gmail/api/guides>;
 - <https://developers.google.com/gmail/api/v1/reference?hl=en>;
- Github <https://developer.github.com/v4/>;
- Météo <https://www.climacell.co/weather-api/docs/>;
- SNCF <https://www.digital.sncf.com/startup/api>;

Nous avons choisi tester l'API Gmail de deux façons. Notre test c'est traduit par les étapes suivantes :

via .NET

1. Lecture du guide et prise en main du vocabulaire destiné à l'API Gmail.
2. Mise en place d'un projet nommé **Quickstart** en .NET.
 - a) Créer un projet via [Google Console Platform](#) et pré-requis sous jacent.
 - b) Créer l'application bureau .NET qui permet de consommer l'API selon le code fourni sur le guide.

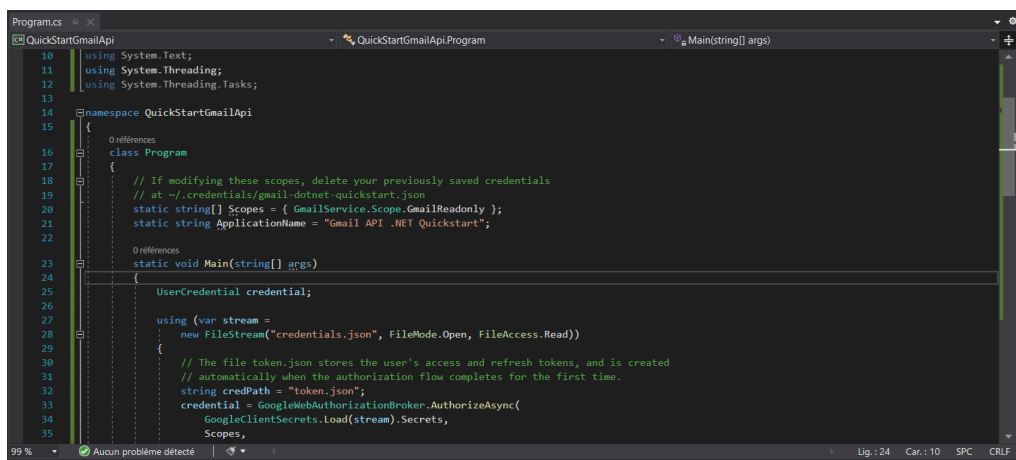


FIGURE 1 – Capture d'écran du code de l'application .NET

via Postman

1. Nous avons repris le même projet GCP défini précédemment.
2. Nous avons suivi [ce tutoriel](#) qui nous indique comment utiliser le protocole d'authentification OAuth 2.0 depuis Postman.

```
C:\Users\DELL\source\repos\QuickStartGmailApi\QuickStartGmailApi\bin\Debug\netcoreapp3.1\QuickStartGmailApi.exe
Credential file saved to: token.json
Labels:
CHAT
SENT
INBOX
IMPORTANT
TRASH
DRAFT
SPAM
CATEGORY_FORUMS
CATEGORY_UPDATES
CATEGORY_PERSONAL
CATEGORY_PROMOTIONS
CATEGORY_SOCIAL
STARRED
UNREAD
Nouveau label pour C#
Nouveau label pour C#/Nouveau label imbriqué pour C#
```

FIGURE 2 – Capture d’écran de l’exécution de l’application .NET

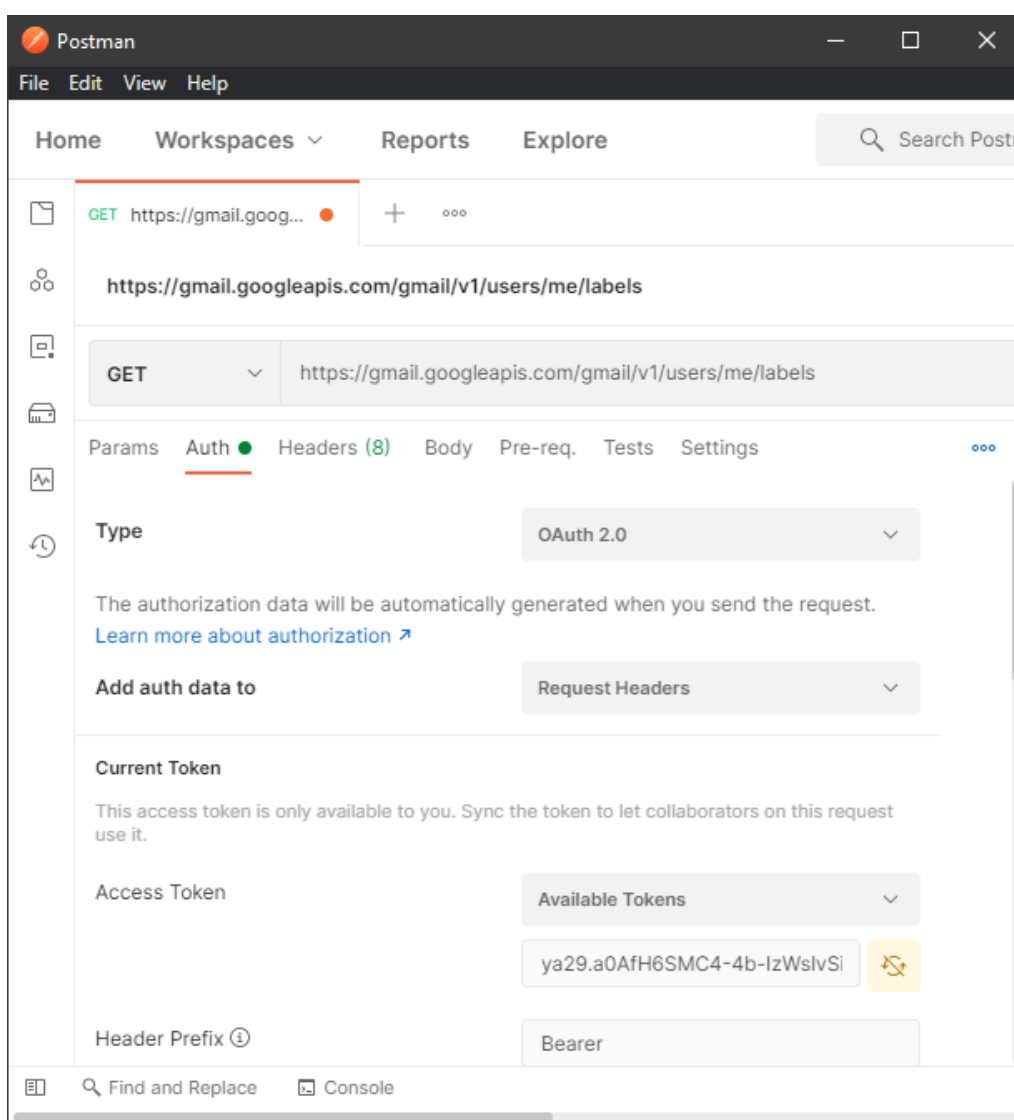


FIGURE 3 – Capture d’écran partielle d’une requête GET sur l’API Gmail

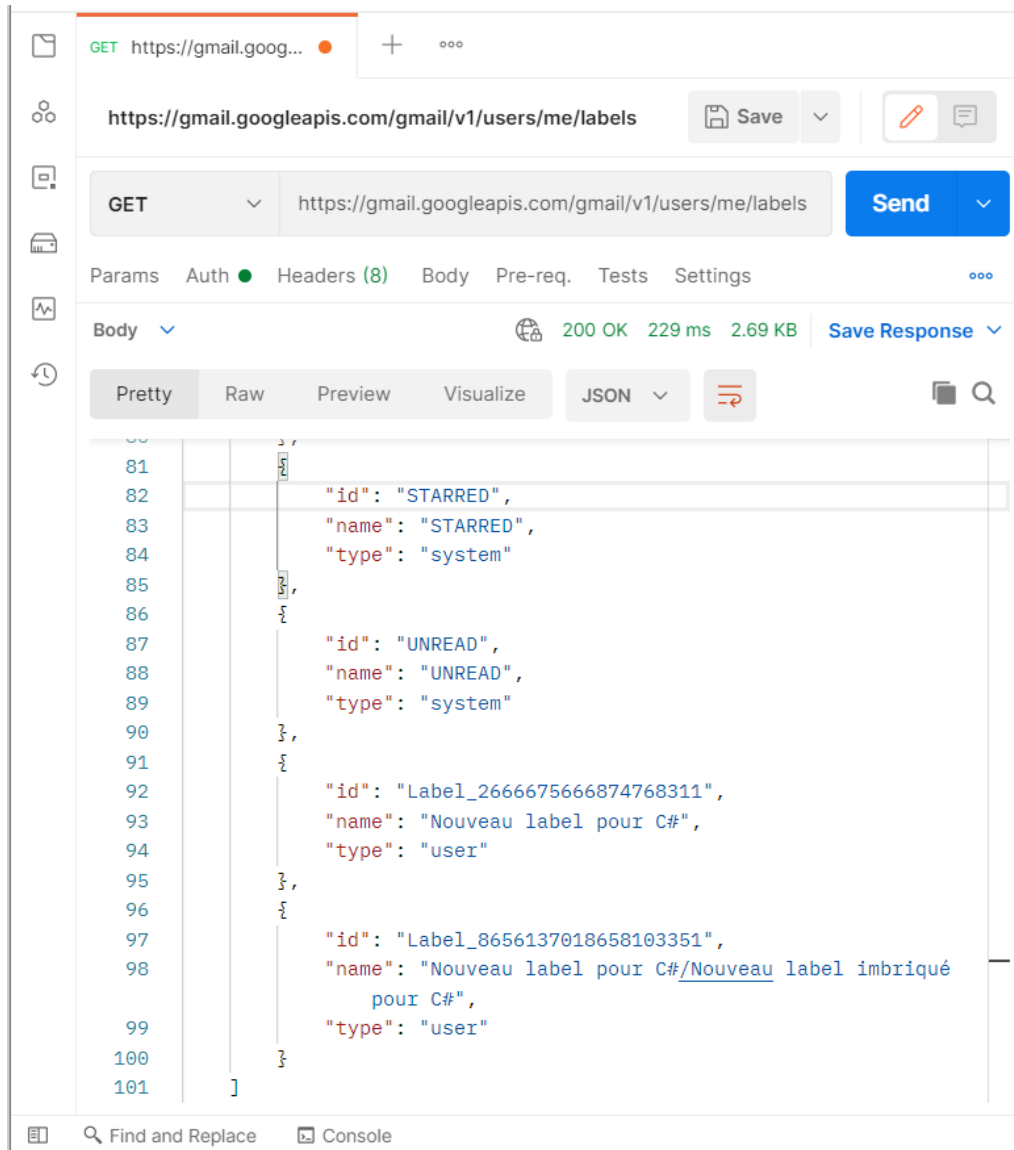


FIGURE 4 – Capture d\u00e9cran partielle d\u2019une requ\u00eate GET sur l\u2019API Gmail

Deuxième partie

Réalisation du TP

2 Création et utilisation de services web

Introduction

Le projet est géré via Git hébergés sur <https://www.gitlab.com>, à l'adresse :

1. <https://gitlab.com/kaciahmed3/gestionhotelsavecwebservicesrest>.

L'adresse mail de notre référent M. Abdelhak-Djamel Seriai fourni au préalable a été ajoutée au dépôt susmentionné :

— rechercheseriai@gmail.com.

2.1 Rappel

Remarque : Ce TP est une suite de celui lié au service web SOAP (même conception métier).

L'objectif de ce TP 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 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'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 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 des cas d'utilisations et diagramme de flux

Nous avons schématisé notre compréhension de la consigne précédente par un diagramme de cas d'utilisation (Voir figure 5) et un diagramme de flux (Voir la figure 6). Cela nous a permis de délimiter les fonctionnalités de notre système, ainsi que le schéma global de sa réalisation.

Par ailleurs, ces diagrammes n'ont pas évolué par rapport à la première conception comme représentés ci-dessous.

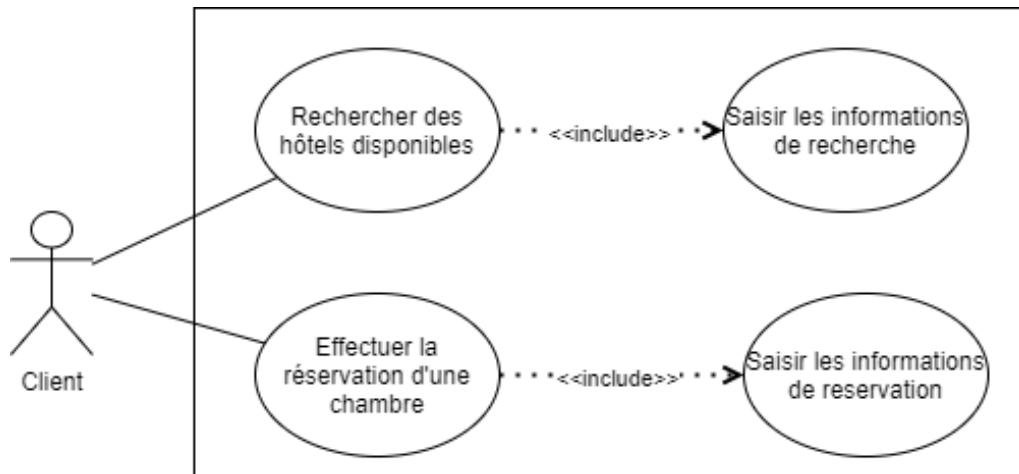


FIGURE 5 – Diagramme UML use case de l'application

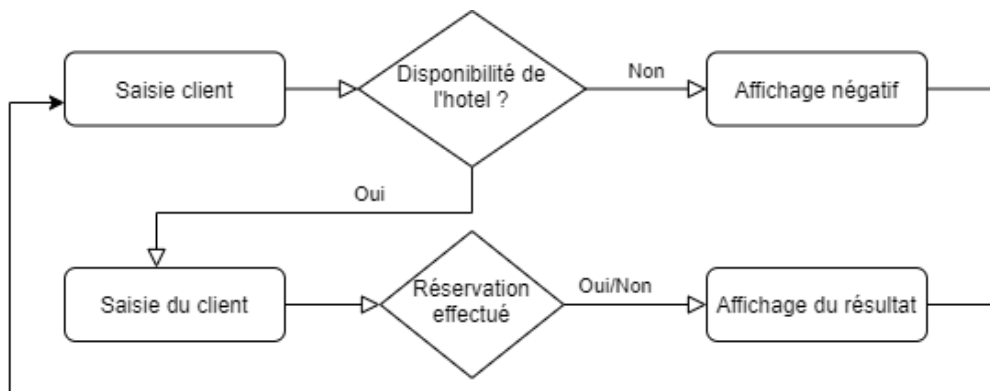


FIGURE 6 – Diagramme de flux de l'application

2.3 Version distribuée-Agence de voyage et hôtels

La première version répondant a la première question est disponible sur ce [dépôt](#).

Définition et présentation de l'architecture mise en place

Grâce aux informations précédentes, nous savons ce que nous allons modéliser grâce aux structures OO ¹ de C#.

Voici le diagramme de classes en figure 7.

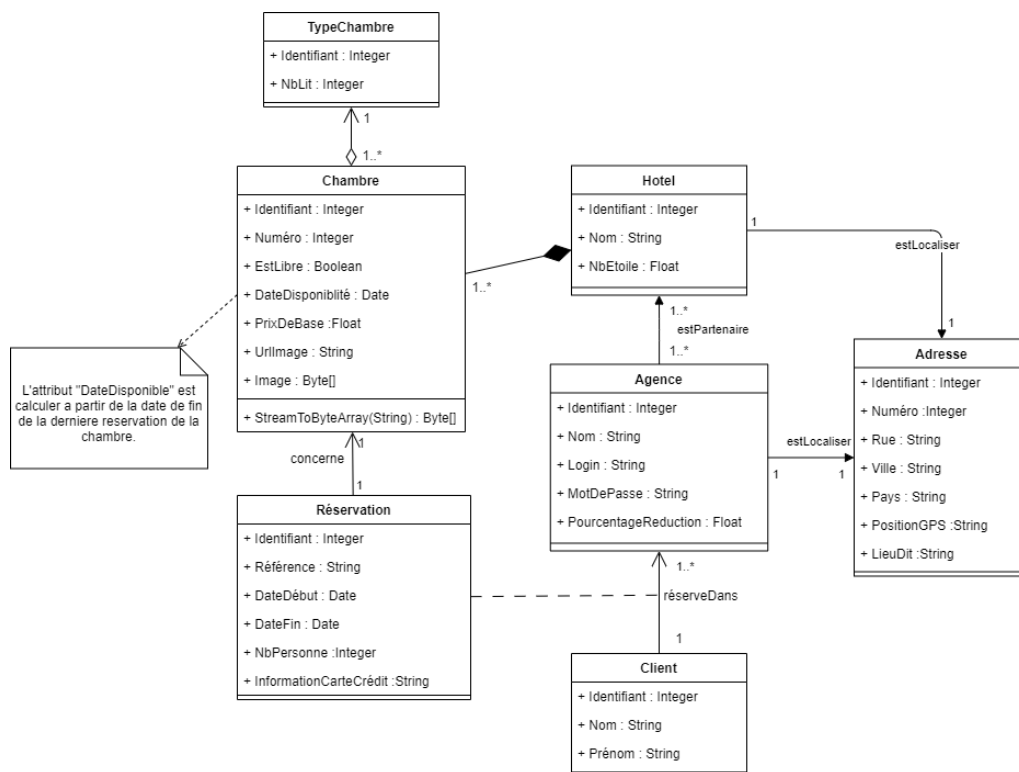


FIGURE 7 – Diagramme de classes de l'application distribué

1. Object Oriented

Description du diagramme de classe

Tout naturellement, nous savons qu'une chambre est caractérisée par un numéro de porte dans son hôtel. De plus, elle est composée d'un certain nombre de lits. En outre, 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 à 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 à héberger, 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 ainsi qu'une agence.

Description de l'architecture de l'application

Comme explicitée dans le sujet du TP, l'architecture de notre application est une architecture distribuée, où le client, qui est représenté par une agence, communique avec un hôtel désignant un serveur composé de deux Web services :

- Le premier Web Service, permet par la communication d'objet *JSON* de renvoyer la liste des offres disponibles, suite à la réception d'une requête http contenant les critères de recherche du client ;
- Le second Web service permet de sauvegarder les informations d'une réservation effectuée par un client et de retourner la référence de réservation.

Cette architecture est illustrée dans la figure ci-dessous :

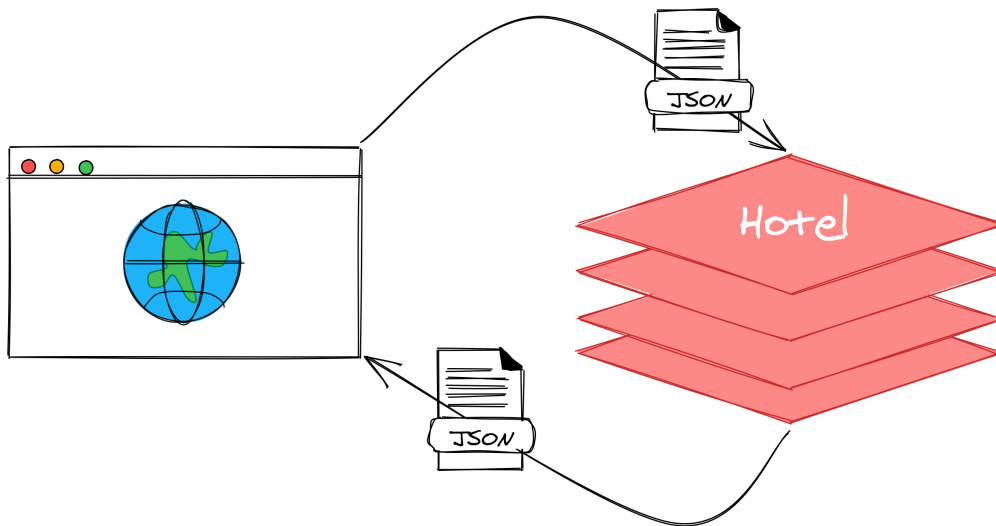


FIGURE 8 – Schéma de l'architecture distribué

En résumé, l'application cliente (disponible à l'agence) communique avec l'API de l'hôtel à travers une première route `GetOffres()` pour chercher la liste des chambres disponibles. En outre, elle utilise une seconde route pour effectuer une réservation d'une certaine offre choisie.

```
C:\Users\DELL\source\repos\gestionHotelsAvecRest\gestionhotelsavecwebservicesrest\Agence\Agence\bin\Debug\Agence.exe
Saisir ville ?
Montpellier
Saisir date d'arriver ?
12-06-2021
Saisir date de depart ?
13-06-2021
Saisir nombre d'étoile ?
5
Saisir nombre de personne a heberger entre 1 et 3 (refaite une reservation pour plus de personne) ?
1
Identifiant = 1_1, Nblit = 1, DateDisponibilite = 01-03-2021,Prix = 46,41
Identifiant = 1_15, Nblit = 1, DateDisponibilite = 10-03-2021,Prix = 37,31
Identifiant = 1_12, Nblit = 1, DateDisponibilite = 10-04-2021,Prix = 62,79
Saisir voulez vous continuer (1 = oui / -1 = non) ?
1
Saisir votre nom ?
kaci
Saisir votre prénom ?
ahmed
Saisir les informations de votre carte de crédit ?
147852369
Veillez sélectionner l'identifiant de l'offre que vous souhaitez, SVP (pour quitter tapez -1)
Saisir identifiant de l'offre ?
1_12
Réservation réussi.
La référence de votre réservation est : "#1945"
Saisir voulez effectuer une autre réservation (1 = oui / -1 = non) ?
1
```

FIGURE 9 – Capture d'écran du processus de recherche et de réservation par la console Agence

2.4 Le service web intègre des images

Le liens suivant [tag](#) vous redirige vers le commit qui met à votre disposition le code répondant à la seconde question.

Explication du travail réalisé

Voici la consigne.

Modifier l'application précédente pour que le service « Service web 1 : Consulter les disponibilités par les agences partenaires » puisse retourner une image de la chambre proposée au client.

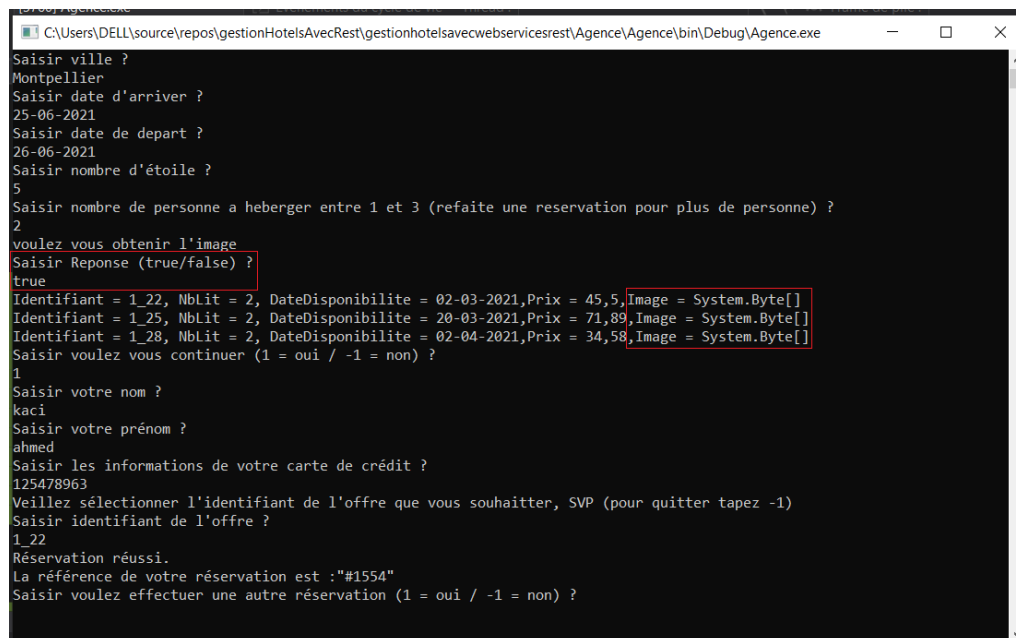
Afin de réaliser cette version, nous avons astucieusement repris la fonction *StreamToByteArray* définis dans notre précédent projet (Rendu pour le TP SOAP) afin de pouvoir calculer la conversion de l'image au format d'une chaîne de caractères ajoutés comme propriété d'une entité Offre.

La fonction introduite ci-dessus, effectue 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.

Le chemin des images étant codé en dur, nous expliquons dans la section 2.9 : Configuration, à quels endroits du code il faudrait le paramétrer pour pouvoir ré-exécuter le projet depuis votre environnement.

Captures d'écran

Il est à noter dans la figure 11, que le nom de l'image est préfixé par l'identifiant de l'offre.



```
C:\Users\DELL\source\repos\gestionHotelsAvecRest\gestionhotelsavecwebservicesrest\Agence\Agence\bin\Debug\Agence.exe
Saisir ville ?
Montpellier
Saisir date d'arriver ?
25-06-2021
Saisir date de depart ?
26-06-2021
Saisir nombre d'étoile ?
5
Saisir nombre de personne a heberger entre 1 et 3 (refaite une reservation pour plus de personne) ?
2
voulez vous obtenir l'image
Saisir Reponse (true/false) ?
true
Identifiant = 1_22, Nblit = 2, DateDisponibilite = 02-03-2021,Prix = 45,5,Image = System.Byte[]
Identifiant = 1_25, Nblit = 2, DateDisponibilite = 20-03-2021,Prix = 71,89,Image = System.Byte[]
Identifiant = 1_28, Nblit = 2, DateDisponibilite = 02-04-2021,Prix = 34,58,Image = System.Byte[]
Saisir voulez vous continuer (1 = oui / -1 = non) ?
1
Saisir votre nom ?
kaci
Saisir votre prénom ?
ahmed
Saisir les informations de votre carte de crédit ?
125478963
Veuillez sélectionner l'identifiant de l'offre que vous souhaitez, SVP (pour quitter tapez -1)
Saisir identifiant de l'offre ?
1_22
Réservation réussie.
La référence de votre réservation est : "#1554"
Saisir voulez effectuer une autre réservation (1 = oui / -1 = non) ?
```

FIGURE 10 – Capture d'écran du processus de sauvegarde de l'image par la console Agence

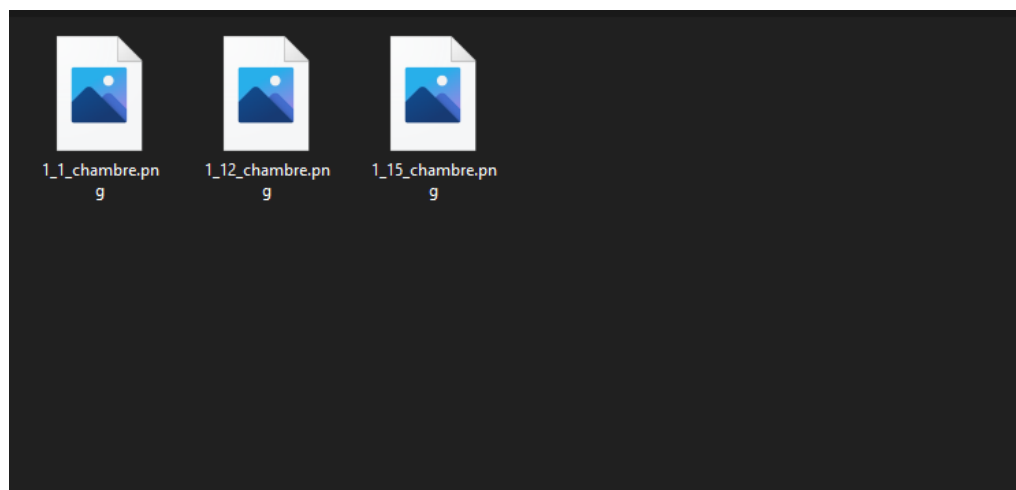


FIGURE 11 – Capture d'écran du résultat de la sauvegarde des images par la console Agence

2.5 Un service web proposé par les agences de voyage pour comparateurs

Voici la consigne,

Nous souhaitons proposer à certaines applications (exemple : Trivago) un service web leur permettant de réaliser la comparaison d'offres d'hébergement d'hôtels proposés par les agences de voyage en ligne.

Pour cela, un client (un usager) fournit à l'application « Comparateur », via une interface IHM (web, console ou autre), les données suivantes concernant son hébergement : ville d'hébergement, dates début et fin de la réservation, nombre de personnes à héberger et le type d'hôtel (nombre minimum d'étoiles).

L'application en question (Trivago) utilisera respectivement le service web proposé par chaque agence de voyage pour obtenir les informations suivantes : une liste d'offres où chaque offre contient les données suivantes : nom d'hôtel, adresse de l'hôtel, nombre d'étoiles de l'hôtel, nombre de lits proposés, prix proposé.

L'application utilisera ces réponses pour répondre à son tour au client/usager (son utilisateur) en lui fournissant les offres reçues avec les noms des agences en ligne qui les proposent.

Face à ce problème, nous avons créé deux autres projets dans la solution de l'Agence nommé respectivement *AgenceAPI* et *AgenceAPI2*. Ainsi qu'une nouvelle solution *Compareur*, incluant le projet *Trivago*. Ces Projets s'inspirent du fonctionnement du projet *Agence* (Console).

Dans le projet Trivago introduit si dessus, nous avons créé une nouvelle classe *OffreCompareur* basé sur la classe *Offre* déjà existante. Dans cette nouvelle classe, nous avons ajouté les propriétés *Nom* et *Adresse* de l'hôtel ainsi que le *Nom* de l'agence en question.

Le comparateur *Trivago*, est considéré comme client de l'agence qui représente, dans ce cas, un serveur. L'agence à son tour, est client du serveur que représente l'hôtel.

La figure 12 suivante illustre le précédent paragraphe :

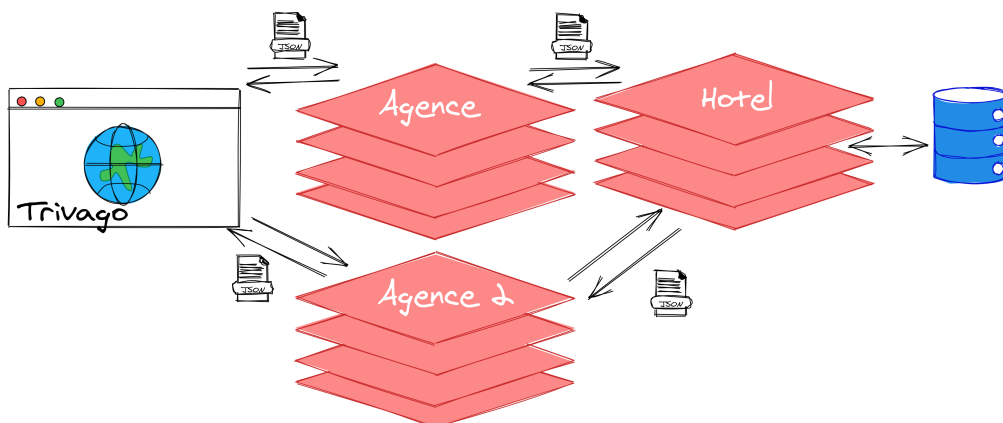


FIGURE 12 – Schéma de l'architecture du comparateur Trivago

Nous avons sauté l'étape de réalisation via la console (pour des contraintes de temps) et directement implémenté une interface graphique (voir figure 13 et 14) sous la technologie *Windows Forms*.

Captures d'écran

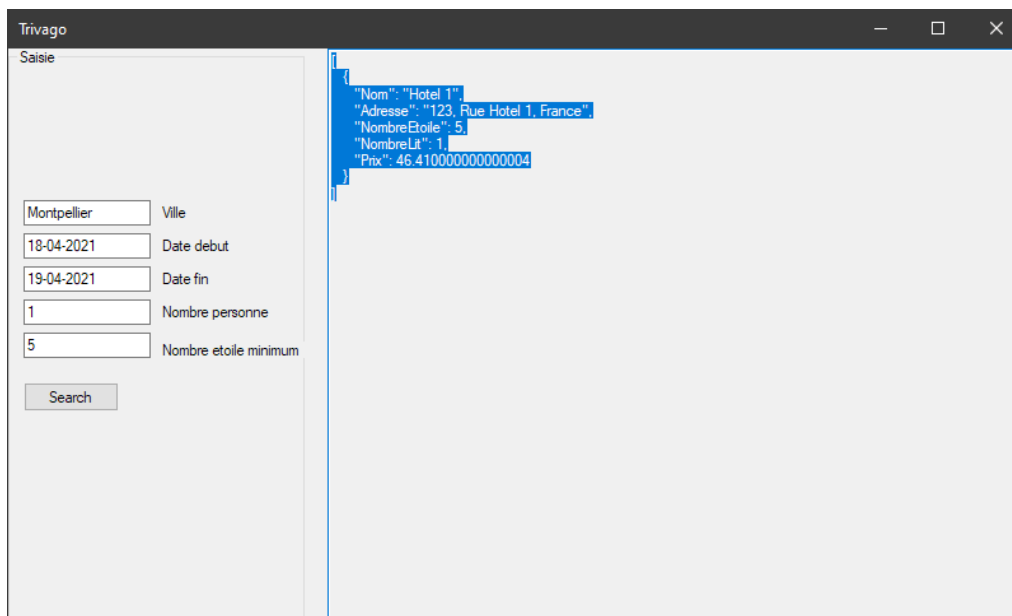


FIGURE 13 – Capture d'écran de l'interface IHM du comparateur

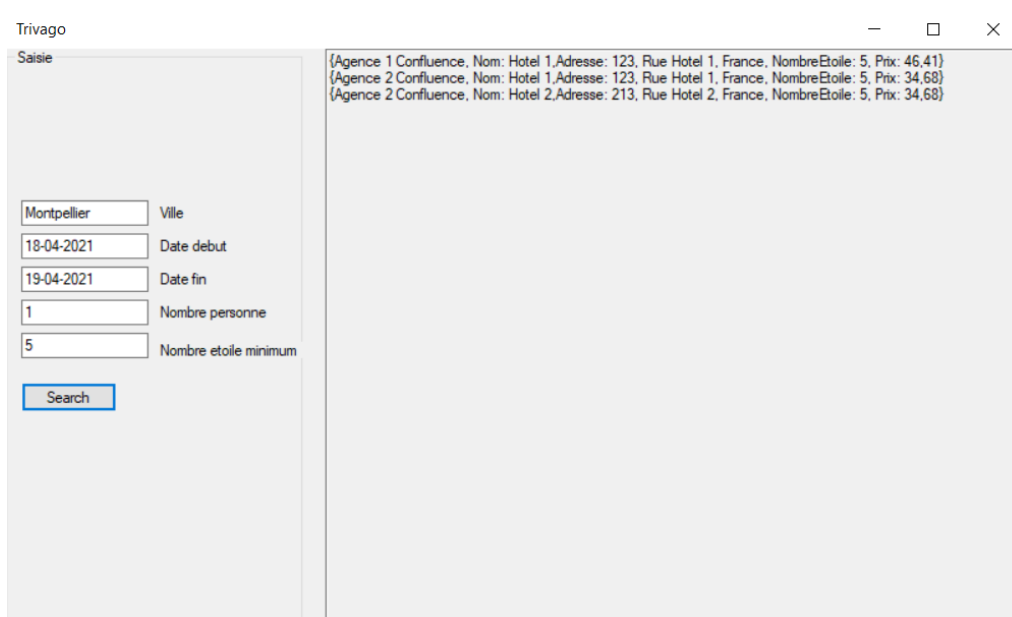


FIGURE 14 – Capture d'écran du résultat de la recherche par le comparateur via plusieurs Agences

2.6 Base de données

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

Dans ce nouveau projet, la [documentation](#) M\$ nous à orienter a utiliser EF6 pour pouvoir gérer nos données. **EF6** nous a permit d'abstraire la gestion des données en séparant les différentes couches de l'application. Ainsi, l'interface graphique d'interaction avec l'application est représentée par la couche `UI`, l'ensemble des controleurs et des modèles utilisé dans les webservice représente la couche métier, et en fin les données manipuler sont représenté par la couche `Donnée`.

la figure suivante illustre de façon schématique ce découpage :

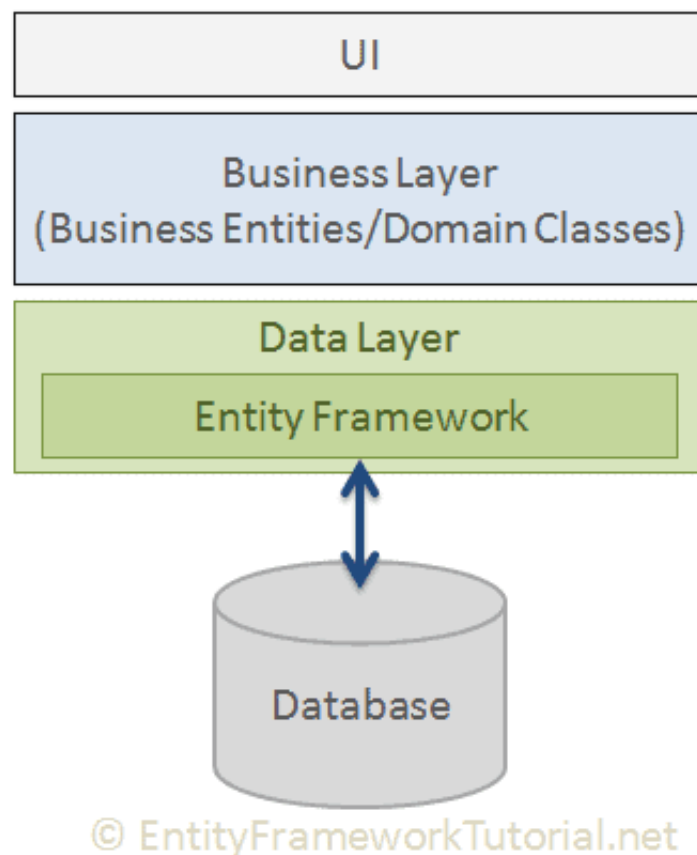


FIGURE 15 – Schéma d'architecture de la gestion des données avec EF6 selon [entityframeworktutorial](#)

Workflow

Nous avons beaucoup appris de l'approche **Code-First** dans l'utilisation du framework EF6 et du Domain-Driven Design.

La prise en main des conventions nous a été fastidieuse, la compréhension des annotations et de l'API Fluent tout autant, cependant nous avons grandement apprécié la lecture de la documentation ainsi que de bon tutoriel comme www.entityframeworktutorial.net.

Par ailleurs, nous avons apprécié le workflow qu'il faut employer pour son utilisation, que nous figurons ci-dessous :

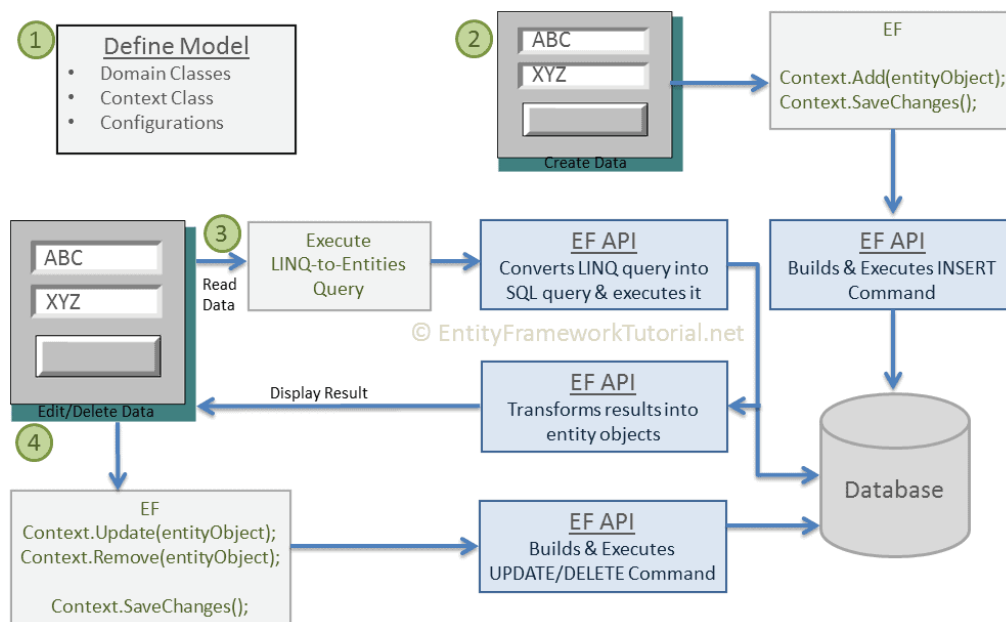


FIGURE 16 – Schéma du workflow avec EF6 selon [entityframeworktutorial](http://entityframeworktutorial.net)

Nous présentons maintenant comment la base de données, s'intègre dans notre application.

Dans cette partie de l'application, nous avons implémenté une interface pour la base de données introduite précédemment. Cette interface permet de récupérer les informations stockées sur l'hôtel, sur la disponibilité des chambres, les agences et les réservations, en communiquant avec la base de données.

Le schéma suivant (figure 17) permet d'illustrer l'architecture globale de notre système.

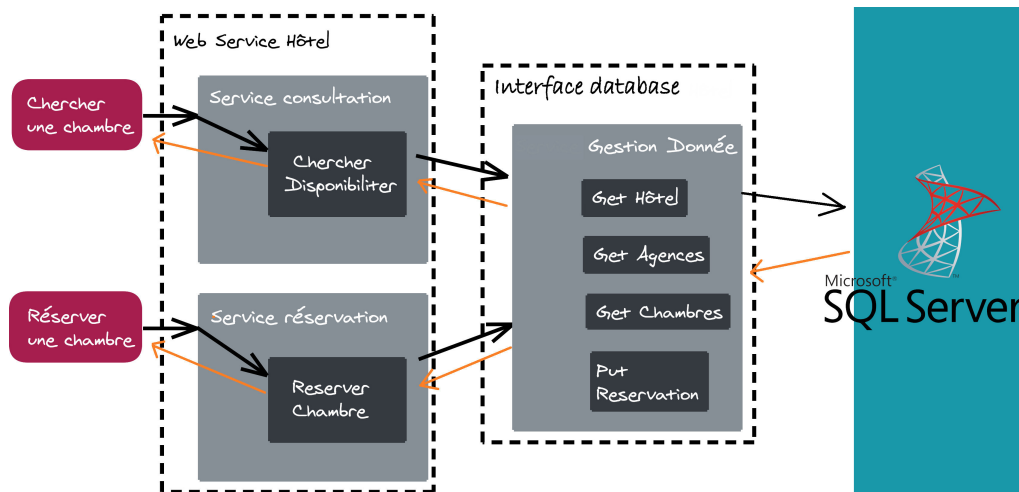


FIGURE 17 – Schéma de l'architecture du système

De ce fait, une base de données M\$ SQL Server Express est utilisée.

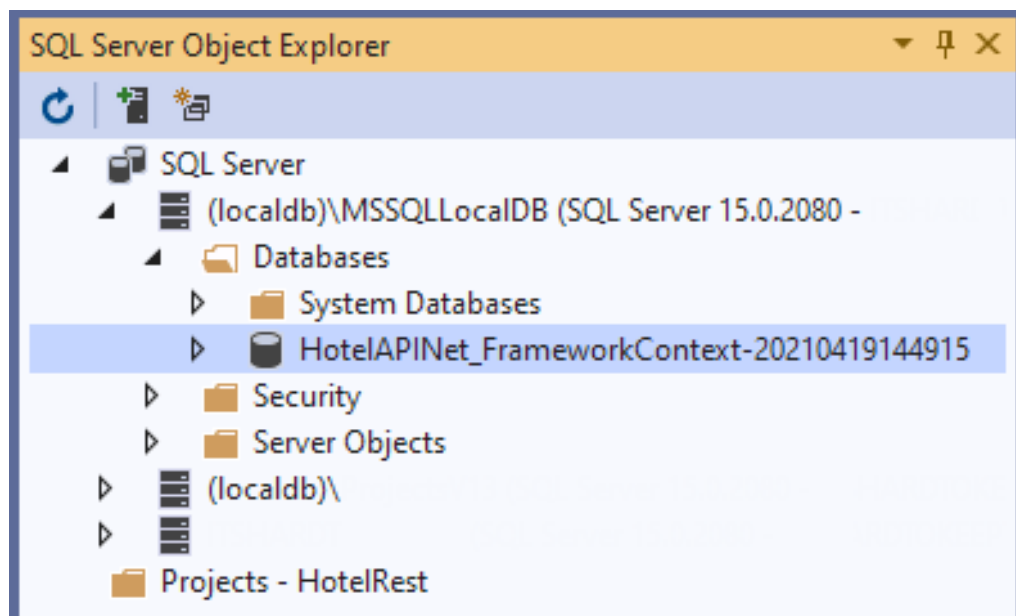


FIGURE 18 – Capture d'écran de Visual Studio de la fenêtre Sql Server

Troisième partie

Conclusion

Première partie

2.7 Tester des services REST

Nous pouvons conclure que le développement et la réalisation de test d'API Web REST est beaucoup plus adapté (même nécessaire !) et facilité par l'utilisation d'outils comme Postman, ou encore built-in comme [Swagger](#) pour pouvoir documenter et tester des API.

Ainsi, afin d'explorer ces technologies nous avons tester l'API *Gmail* avec Postman et via une application .NET.

Deuxième partie

2.8 Création et consommation de service REST

Tout bien considéré, l'architecture métier (logique) cœur de notre application n'a pas bougé par rapport à celle étudiée pour le TP SOAP.

La façon de communiquer et de gérer une requête est certes différente. Cependant l'utilisation de framework, encapsulant dans des abstractions de plus haut niveau, comme [HttpClient](#) à la place de [HttpWebRequest](#) permettent de s'affranchir de tels problèmes d'implémentation.

Nous observons systématiquement que les opérations d'un contrôleur sont implémentées avec la conception *CRUD* comme objectif. Un tel objectif permet d'éviter des erreurs de conception et de fournir une interface claire, nette et précise pour les développeurs interne/externe.

Page blanche

2.9 Configuration

Le dossier *assets* contient les images des chambres, le fichier *Configuration.cs* permet d'insérer les données de tests dans la BDD.

Nous décrivons succinctement les deux configurations possible :

1. [Version avec images](#)

- L'introduction des images par le web service, contraint l'hébergeur de l'API Hotel a modifier les chemin vers les images qui est écrit en dur dans le fichier *gestion-hotelsavecwebservicest/HotelRest/HotelRest/Controllers/MaDatabase.cs*. Il faudra modifier la variable de classe **CHEMIN_DOSSIER_ASSETS** qui devra mener jusqu'à la racine du dossier asset contenu dans le projet.

Inspirez-vous du chemin déjà utilisé.

2. L'introduction d'un SGBD par l'API, délocalise l'écriture en dur du chemin vers les images dans la base de données.

Il faudra modifier la variable de classe **CHEMIN_DOSSIER_ASSETS** servant aux insertions SQL de la table Chambre, contenu dans le fichier *gestion-hotelsavecwebservicest/HotelRest/HotelRest/Migrations/Configuration.cs*.

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.