Kwitonda, Fabrice

Fabricekfr@gmail.com

Processus cs-02  
Hackathon

Système de gestion de réservations

Table des matières

[Aperçue de l’architecture 2](#_Toc516577498)

[Problématique 2](#_Toc516577499)

[Fonctionnalités et contraintes 2](#_Toc516577500)

[Solution proposée 2](#_Toc516577501)

[**Exigences rencontrées** 2](#_Toc516577502)

[Vue conceptuelle 3](#_Toc516577503)

[Vue logique 3](#_Toc516577504)

[Modèle conceptuel de données 4](#_Toc516577505)

[Diagrammes de classes 4](#_Toc516577506)

[WCFService 4](#_Toc516577507)

[DataAccess 5](#_Toc516577508)

[Domain 5](#_Toc516577509)

[Utilisation 6](#_Toc516577510)

[Serveur Windows/IIS 6](#_Toc516577511)

[Dans l’enivrement Visual Studio 6](#_Toc516577512)

[AWS Elastic Beanstalk 6](#_Toc516577513)

[Tâches effectuées lors de l’initialisation de l’application 6](#_Toc516577514)

[Annexes 7](#_Toc516577515)

[Script de création de la base de données SQLite 7](#_Toc516577516)

[Réservations initiales 8](#_Toc516577517)

# Aperçue de l’architecture

## Problématique

Transport Canada (TC) est à la recherche d’un système de réservation pour les services offerts. Puisque les centres sont situés partout au Canada, le système doit avoir une base de données centrale pour gérer tous les rendez-vous. De cette façon, le commis pourra effectuer une réservation pour un service à partir de n’importe quelle région.

## Fonctionnalités et contraintes

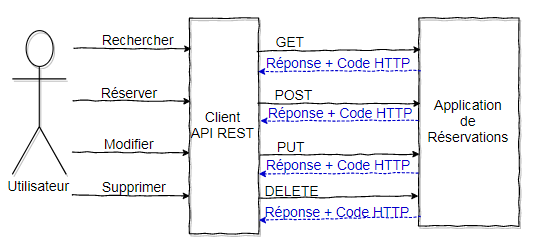
1. L’application doit être une **API-REST** (interface de programmation applicative)
2. L’application doit permettre à une application interne de créer, rechercher ou annuler une réservation.
3. L’application n’a pas besoin d’authentification ou d’autorisation.
4. L’application retournée des réponses en format JSON
5. Un centre de service peut seulement effectuer une réservation par jour.
6. Il existe différents types de centres de services de TC

# Solution proposée

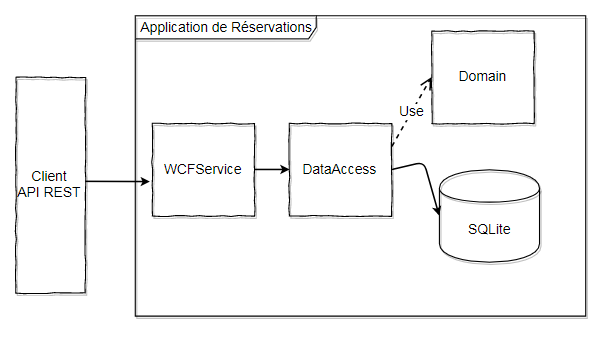
**Exigences rencontrées**

1. Une solution **WCF** (Windows Communication Foundation)
2. Développée à l’aide du logiciel Visual Studio 2015 (C#, .Net Framework 4.5.2)
3. Déployable sur les machines Windows
4. Ne nécessitant pas d’authentification ou d’autorisation
5. Permettre une recherche, mise à jour et suppression de réservations en appelant les méthodes de requête HTTP (GET, PUT, POST, DELETE)
6. Retourne les réponses en format JSON
7. Ne pas permettre pas plus d’une réservation pour un même jour à un même centre

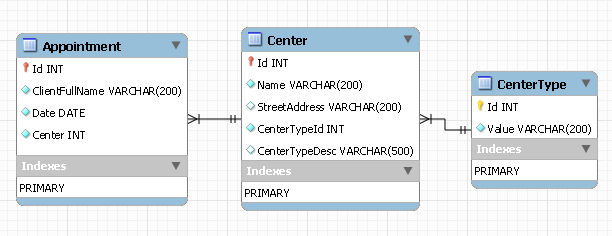
## Vue conceptuelle



## Vue logique



## Modèle conceptuel de données

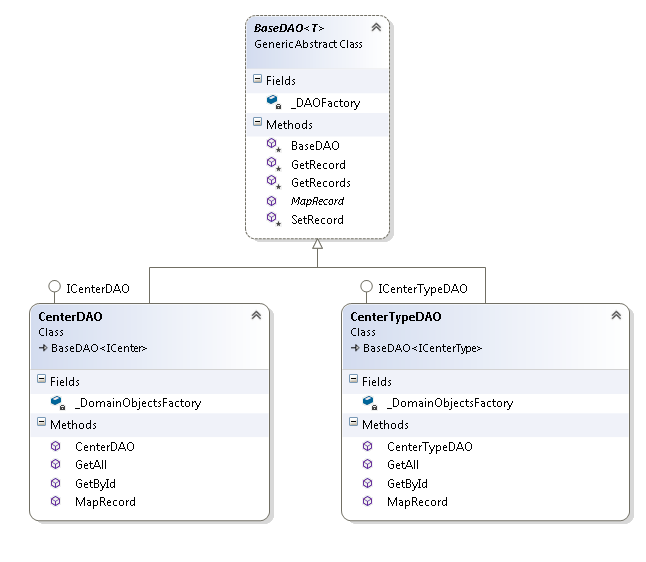


## Diagrammes de classes

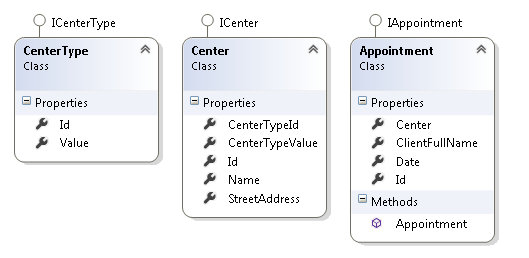
### WCFService



### DataAccess



### Domain



# Utilisation

## Serveur Windows/IIS

1. Extraire le contenu du fichier « TCServiceCenterBookingApplication.zip »
2. Ouvrir la solution « TCServiceCenterBookingApplication.sln » dans Visual Studio
3. Recompiler la solution « Rebuild solution »

On doit avoir accès à l'internet pour que Visual studio importer les DLLs nécessaires

1. Ouvrir le menu contextuel sur le projet « WCFService »
2. Publier sur le serveur de votre choix

## Dans l’enivrement Visual Studio

1. Extraire le contenu du fichier « TCServiceCenterBookingApplication.zip »
2. Ouvrir la solution « TCServiceCenterBookingApplication.sln » dans Visual Studio
3. Recompiler la solution « Rebuild solution »

On doit avoir accès à l'internet pour que Visual studio importer les dll nécessaires

1. Ouvrir le menu contextuel sur le fichier BookingApplicationService.svc du projet « WCFService »
2. Choisir et cliquer « Voir dans le navigateur »
3. L’application sera accessible à l’adresse « http://localhost:55001/BookingApplicationService.svc »

## AWS Elastic Beanstalk

L’application a été déjà déployée sur le service d’infonuagique AWS Elastic Beanstalk à l’adresse

<http://fkservicecenterbooking-test.us-east-2.elasticbeanstalk.com/BookingApplicationService.svc>

## Tâches effectuées lors de l’initialisation de l’application

1. Création de la base de données à partir du script de la première annexe
2. Importation des données du fichier Centers.json
3. Ajout de deux réservations selon la deuxième annexe

# Annexes

## Script de création de la base de données SQLite

-- MySQL Script generated by MySQL Workbench

-- Wed Jun 6 13:51:41 2018

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema TCDatabase

-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `TCDatabase` ;

-- -----------------------------------------------------

-- Schema TCDatabase

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `TCDatabase` DEFAULT CHARACTER SET utf8 ;

USE `TCDatabase` ;

-- -----------------------------------------------------

-- Table `TCDatabase`.`CenterType`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `TCDatabase`.`CenterType` ;

CREATE TABLE IF NOT EXISTS `TCDatabase`.`CenterType` (

`Id` INT NOT NULL AUTO\_INCREMENT,

`Value` VARCHAR(200) NOT NULL,

PRIMARY KEY (`Id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `TCDatabase`.`Center`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `TCDatabase`.`Center` ;

CREATE TABLE IF NOT EXISTS `TCDatabase`.`Center` (

`Id` INT NOT NULL AUTO\_INCREMENT,

`Name` VARCHAR(200) NOT NULL,

`StreetAddress` VARCHAR(200) NULL,

`CenterTypeId` INT NOT NULL,

`CenterTypeDesc` VARCHAR(500) NULL,

PRIMARY KEY (`Id`),

CONSTRAINT `fk\_Center\_CenterType\_Id`

FOREIGN KEY (`Id`)

REFERENCES `TCDatabase`.`CenterType` (`Id`)

ON DELETE RESTRICT

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `TCDatabase`.`Appointment`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `TCDatabase`.`Appointment` ;

CREATE TABLE IF NOT EXISTS `TCDatabase`.`Appointment` (

`Id` INT NOT NULL AUTO\_INCREMENT,

`ClientFullName` VARCHAR(200) NOT NULL,

`Date` DATE NOT NULL,

`Center` INT NOT NULL,

PRIMARY KEY (`Id`),

CONSTRAINT `fk\_Appointment\_Center\_Id`

FOREIGN KEY (`Id`)

REFERENCES `TCDatabase`.`Center` (`Id`)

ON DELETE RESTRICT

ON UPDATE CASCADE)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

## Réservations initiales

[

{"Id": 1,

"ClientFullName": "John Doe",

"Date": "2018-01-01",

"Center": {"Id": 4,

"Name": "Dartmouth Transport Canada Centre",

"StreetAddress": "Office, Queen Square Darthmouth

"CenterTypeValue": "Transport Canada owned and operated / Détenu et exploité par ransport Canada"}

},

{"Id": 2,

"ClientFullName": "Martine Duplessi",

"Date": "2018-11-07",

"Center": {"Id": 55,

"Name": "Bobbi Stout",

"StreetAddress": "Coastal Pacific 30575 Approach Drive

"CenterTypeValue": "Authorized Examination Invigilators /Surveillant d'examen autorisés"}

},

]