Projet final

# Présentation

Votre client « Grand Hôtel » vous commande une application web pour la réservation de chambres par ses clients, et pour des besoins de gestion interne.

Il possède déjà une base de données SQL Server, dont il vous fournit une sauvegarde, ainsi que le modèle physique de données :



Ci-dessous la description des tables et règles de gestions principales :

|  |  |
| --- | --- |
| **Table** | **Description** |
| Client | Un client ne peut avoir qu’une seule adresse, mais plusieurs numéros de téléphones et plusieurs adresses email. |
| Adresse | Adresse unique du client. Certains clients n’ont pas d’adresse.  Lorsqu’elle existe l’adresse est identifiée par l’id du client. |
| Téléphone | Donne la liste des N° de tel. de chaque client. Le champ CodeType vaut M pour un N° de mobile et F pour un N° fixe.  Pro indique s’il s’agit d’un N° professionnel |
| Facture | La date de paiement de la facture est nulle tant que la facture n’est pas payée |
| LigneFacture | Détaille les prix et quantités des services facturés (chambres, petits déjeuner…etc.) pour chaque facture. Une ligne est identifiée par l’id de la facture et un numéro. |
| Réservation | Contient une ligne pour chaque jour et chambre occupée.  Le champ HeureArrivee indique l’heure d’arrivée approximative prévue par le client. La précision est à l’heure près.  Le champ Travail indique si le client réserve pour son travail  Une chambre peut être réservée par plusieurs personnes (ex : un couple avec enfants) à la même date. Mais la réservation correspondante ne peut être reliée qu’à un seul client (qui représente les personnes). |
| Calendrier | Contient tous les jours de chaque année d’exploitation de l’hôtel. Elle est utile pour déterminer facilement le nombre de jours dans chaque mois, ainsi que les jours durant lesquels certaines chambres ne sont pas occupées. |
| Tarif | Contient l’évolution des tarifs des différentes prestations (chambres de types 1 et 2, petit déjeuner…).  Les tarifs sont révisés au 1er janvier de chaque année. |
| TarifChambre | Permet d’associer les chambres aux tarifs. |

Votre client souhaite que l’application puisse tourner sur différents systèmes d’exploitation afin de se donner la possibilité de changer de système si besoin. Une étude préalable réalisée avec lui vous a fait choisir la technologie ASP.Net Core MVC.

Votre client a défini une bonne partie de son cahier des charges, mais certains éléments, comme la définition des rôles ou la gestion du paiement, ne sont pas encore totalement déterminés et ne sont donc pas à gérer pour l’instant. Il souhaite que le visuel et l’ergonomie de l’application soient soignés pour une bonne expérience utilisateur, et vous laisse une certaine liberté dans ce domaine.

L’application sera nommée GrandHotel et devra répondre aux besoins fonctionnels décrits ci-dessous.

# Fonctionnalités de l’application

## Réservation de chambres

Le processus de réservation d’une chambre par un client est le suivant :

1. Le client saisit :

* La date de début de son séjour
* Le nombre de nuits
* Le nombre de personnes
* Son heure d’arrivée prévue
* Si c’est un séjour pour son travail ou non (facultatif)

1. L’application détermine s’il y a des chambres libres qui satisfont la demande du client
2. Si c’est le cas, elle affiche le tarif journalier et le prix total du séjour, et demande au client de confirmer sa réservation. Pour rappel : les tarifs des chambres évoluent au cours du temps.
3. Si le client confirme, il doit s’authentifier ou bien créer un compte, puis arrive sur une page qui récapitule sa réservation et le remercie d’avoir choisi Grand Hôtel.

La gestion du paiement n’est pas demandée pour le moment.

## Gestion du compte client

### Création et modification du compte

Un client crée un compte en renseignant une adresse mail et un mot de passe. Ce dernier doit faire au minimum six caractères et contenir des majuscules et minuscules.

Après création du compte, le client est invité à renseigner ses coordonnées : civilité, nom, prénom, adresse postale, numéros de téléphone (portable et/ou fixe). L’adresse postale est facultative et un seul des N° de téléphone est obligatoire. Le format des N° doit être vérifié.

Ces données sont modifiables depuis un sous-menu du menu « Mon compte » accessible à l’utilisateur authentifié.

### Consultation des factures

Un client doit pouvoir consulter l’historique de ses factures, et accéder au détail de chacune d’elles.

Les factures sont affichées par année et triées par date décroissante pour l’année sélectionnée. Par défaut c’est l’année en cours qui est sélectionnée.

Cette page est également accessible depuis un sous-menu du menu « Mon compte » accessible à l’utilisateur authentifié.

## Gestion de l’hôtel

**Pages spécifiques dans l’application web**

L’application Web doit contenir les pages suivantes, destinées à la gestion interne de l’hôtel, et accessibles par un menu « Gestion interne » :

* Liste des clients (id, nom, prénom, email) avec leurs nombres totaux de réservations effectuées, et s’ils ont ou non une réservation en cours. La liste étant potentiellement grande, on affichera les clients selon la première lettre de leur nom.
* Liste des chambres, avec leur tarif en cours et leur état d’occupation à la date du jour. On doit pouvoir filtrer les chambres selon leur état d’occupation, mais par défaut toutes les chambres doivent être affichées.

NB/ Les différents rôles des utilisateurs de l’application n’étant pas encore définis par votre client, les entrées du menu Gestion interne seront accessibles pour l’instant à n’importe quel utilisateur authentifié.

**API Web**

Votre client souhaite pouvoir à terme partager sa liste de clients avec d’autres applications de bureau ou mobiles. Il vous demande donc de créer une API Web permettant de :

* Créer un client (sans son adresse postale, ni ses N° de téléphone pour le moment)
* Rechercher des clients dont le nom contient un texte donné (au minimum trois lettres)
* Récupérer les infos d’un client identifié par son id : infos du client, adresse postale, N° de téléphone

Il souhaite que vous lui fournissiez une petite application console permettant de tester cette API Web.

## Recommandations techniques

L’application doit gérer l’authentification par comptes d’utilisateurs individuels. Les données correspondantes seront stockées dans des tables spécifiques à ajouter dans la base au moyen d’une migration. On utilisera pour cela la commande « update-database -context ApplicationDbContext » dans le gestionnaire de package NuGet.

Pour faire la correspondance entre un client de la table Client et un compte utilisateur de la table AspNetUsers, on utilisera l’adresse email du client.

Si besoin, vous pouvez insérer des données supplémentaires dans la base pour pouvoir tester certaines fonctionnalités. Dans ce cas, pensez à faire une sauvegarde de la base pour votre présentation, et fournissez-là avec vos sources.

# Livraison

Vous devez fournir :

* L’ensemble des sources en fichier zip et sur GitHub
* Une sauvegarde de la base si vous avez modifié les données
* Une doc technique décrivant les principaux choix techniques que vous avez faits en termes d’architecture, de découpage du code, d’algorithmes, d’impact sur les performances de l’application...etc. Cette doc peut aussi mentionner les points sur lesquels vous pensez que votre code n’est pas optimal et les améliorations possibles. Elle permet notamment de mesurer votre capacité d’analyse, votre capacité à identifier les points essentiels, et le recul que vous avez sur ce que vous faites.

**Important :** ajoutez votre nom en commentaire au-dessus de chaque méthode, afin que je puisse mieux évaluer votre qualité de codage individuelle.

# Critères d’évaluation

|  |  |
| --- | --- |
| **Livraison** | **Fonctionnel et ergonomie** |
| L'ensemble des éléments demandés est fourni (source, sauvegarde de base, doc)  Le code compile et s'exécute correctement avec la base fournie | L'appli fournit les fonctionnalités demandées avec un bon niveau de fiabilité  L'ergonomie des pages répond aux besoins et permet une utilisation simple de l'appli  L'interface visuelle est soignée  Des contrôles de validation sont en place |
| **Architecture technique** | **Qualité du code** |
| Les principes d'architecture sont respectés.  La doc technique est claire.  Les choix techniques sont décrits, justifiés et pertinents | La sémantique est pertinente Les conventions de codage sont respectées  Le code est commenté et facile à lire  La solution Visual Studio est bien organisée |