**Projet de développement pour l’examen: RentAcar**

Table des matières

[**Enoncé** 2](#_Toc95151527)

[Les contraintes techniques : 2](#_Toc95151528)

[Remarques : 3](#_Toc95151529)

[**Schéma Entité-Association** 4](#_Toc95151530)

[Règles 4](#_Toc95151531)

[Choix 4](#_Toc95151532)

# **Enoncé**

C’est une société qui gère des leasings de voiture…

Evidemment, il y plusieurs voitures immatriculées, des type différents et une notoriété adéquate.

Cette société opère dans plusieurs pays et villes dans ceux-ci.

Chaque voiture est liée à un bureau/dépôt.

Un client peut réserver une voiture et définir son voyage. Ce dernier est constitué d’une date de départ et de fin et d’un bureau de départ et du type de calcul de la course.

Si le client réserve x semaines avant son départ, il a droit a une réduction de 5 %/semaine jusqu’à un maximum de 20%. La dernière semaine n’a pas de réduction possible.

Le prix de la location de la voiture est en fonction de sa notoriété (Luxe, Van,…).

Le prix de la course est calculé

* Soit en fonction du dépôt de retour de la voiture. Si le retour est défini au moment de la location, une ristourne de 5% est appliquée. Mais si le dépôt de retour n’est pas celui défini, une amende de 10 % luis sera appliquée au forfait le plus onéreux.
* Soit en fonction du kilométrage effectué et le prix du kilométrage est lié au pays auquel appartient le dépôt de départ.

Il existe donc une liste des forfaits entre les dépôts. On ne peut pas proposer un forfait s’il n’existe pas et donc le prix au KM devra être choisi si nécessaire.

Le retour d’une voiture se fait dans le dépôt et ce dernier retrouve le contrat de location y indique soit le dépôt d’arrivée soit le kilométrage adéquat et fournit la facture au client.

2 applications sont nécessaires…

Une application pour le client qui fait la réservation et l’administration.

Une application pour le dépôt…

## Les contraintes techniques :

BD de type SQL serveur **+ Pas d’accès aux tables en direct** = SP et schéma … ?

Une découpe en couche + Web API depot et RentaCar (Client)

Une application desktop au minimum permettra au client de faire sa réservation et la suivre.

Une application desktop au minimum permettra de gérer les voitures, les dépôts, les contrats…

On peut choisir une autre DB ou langage de dev à condition que les contraintes soient respectées.

## Remarques :

Une voiture ne peut être loué qu’à une seule réservation en même temps.

On ne garde pas l’historique du prix du KM/Pays. Cela vaut dire qu’il faudra garder cette information dans les informations définitives de la réservation.

Pour un moment T1, le prix pour la Belgique est de 3 euros le KM

Une réservation1 est réalisée et calculée en fonction du KM = 100 = 300 au total à payer.

Pour un moment T2, le prix pour la Belgique est de 5 euros le KM

Une réservation2 est réalisée et calculée en fonction du KM = 100 = 500 au total à payer.

Si j’affiche la facture ou les informations de la réservation1, il faudra bien avoir le prix de 3 euros et donc il faudra copier ce prix dans la réservation.

Si pas d’historique dans le pays, dans la table Pays, j’aurai un id, le libellé et le prix au KM. Et on gardera le prix au km dans la réservation au final.

Si on a un historique le pays, j’aurai un id, le libellé et le prix au KM mais aussi une date de validité de départ et de fin (1/1/2021 au 16 Juin 2021). On ne garde pas le prix au km dans la réservation puisque je peux la retrouver grâce à la date de celle-ci.

Forfait entre dépôts…

Si j’ai 3 dépôts comme BXL, Liège et Mons, je peux mettre un forfait pour BXL-Liège, Mons-Liège, Mons-BXL

# **Schéma Entité-Association**

Esquisse voir Draw.Io

## Règles

Prix de la course =

* Base (notoriété) x coefficient multiplicateur du forfait.
* Base(notoriété) x coefficient multiplicateur au kilomètre x nbre de kilomètre

Où

* Forfait x coefficient multiplicateur (Base/notoriété).
* Nombre kilomètre x Coefficient multiplicateur (Base)

=> Contrainte : une valeur absolue dans une des tables ; un coefficient multiplicateur dans l’autre !!!

## Choix

* Historique du prix : nous choisissons de garder un historique des prix via une table « Prix » (voir énoncé).
* Prix : Nous choisissons la solution suivante :
  + Base (notoriété) x coefficient multiplicateur du forfait.
  + Base(notoriété) x coefficient multiplicateur au kilomètre x nbre de kilomètre