**Plan de programmation**

Plan établi en partant de la couche la plus basse (DB) vers la plus haute (interface(s) utilisateur(s)).

Table des matières

[**Architecture** 1](#_Toc96008253)

[Base de données (SQL Server) 1](#_Toc96008254)

[Application - Partie ASP.net WepApi 1](#_Toc96008255)

[*Couche DAL* 1](#_Toc96008256)

[*Couche BL* 1](#_Toc96008257)

[*Couche WebApi* 2](#_Toc96008258)

[*Modèles* 2](#_Toc96008259)

[*Erreurs* 2](#_Toc96008260)

[Application – Partie MVC 2](#_Toc96008261)

[**Programmation** 2](#_Toc96008262)

[Partie WebApi 2](#_Toc96008263)

[Partie MVC 3](#_Toc96008264)

# **Architecture**

## Base de données (SQL Server)

Voir Partie Contraintes + Schéma EA + Schéma relationnel

## Application - Partie ASP.net WepApi

### *Couche DAL*

C’est la couche d’accès aux données. Elle a la charge de réaliser les opérations d’insertion, de suppression et de mise à jour des entités dans la base de données.

Elle est réalisée avec entity Framework. La couche d’accès aux données est donc sous-jacente

Connectée à la DB via la configuration EntityFramework

Est référencée par la BL ; référence les modèles et les erreurs

« SaveChange » sur la DB

### *Couche BL*

C’est la couche qui contient la logique business.

Elle reçoit les instructions et les données depuis la couche WebApi

Elle référence la couche DAL, les modèles et les erreurs.

Elle fait appel à la DAL pour sauvegarder les changements et mise à jour réalisés sur les entités (modèles)

### *Couche WebApi*

C’est la couche qui communique avec la partie MVC via des requêtes HTTP.

Elle reçoit des requêtes du contrôleur MVC et lui renvoie des informations demandées. Elle transmet les requêtes reçues à la BL.

Elle référence la BL, les Erreurs et les modèles.

### *Modèles*

### *Erreurs*

## Application – Partie MVC

### *Contrôleur*

### *Modèles*

### *Vue*

# **Programmation**

## Partie WebApi

* **Modèles**

Entity Framework => Scaffolding =>

Context = RentACarContext

Entities = Voiture ; Client ; Réservation ; etc… (voir Schéma Relationnel)

* **DAL**
  + Classes SaveChange – Disconnected State

Classes qui servent à réaliser les Insert, Update, Delete dans la DB via des méthodes Savechanges

using (var context = new RentaCarContext())

{

//1. Attach an entity to context with Added EntityState

context.Add<Student>(std);

//or the followings are also valid

// context.Students.Add(std);

// context.Entry<Student>(std).State = EntityState.Added;

// context.Attach<Student>(std);

//2. Calling SaveChanges to insert a new record into Students table

context.SaveChanges();

}

<https://www.entityframeworktutorial.net/efcore/saving-data-in-disconnected-scenario-in-ef-core.aspx>

* **BL**
  + Classes interactions client
    - Méthodes de Création/Maj entités
      * Nouveau Client
      * Nouvelle\_réservation
      * Modification\_Réservation ? (changement de véhicule, de dépôt ?)
      * Annulation\_réservation ?
    - Méthodes de calcul
      * Calcul Prix
    - Méthode « Sauvegarde/Confirmation » qui appelle la DAL
  + Classes interactions Loueur
    - Méthodes de Création/Maj entités
      * Mise à jour réservation (date\_Retour, Dépôt\_retour
      * Nouvelle Voiture, MAJ\_Voiture, Suprression\_Voiture
      * Nouveau Prix, MAJ\_Prix, Suppression Prix
      * Nouveau Dépôt, MAJ\_Dépot, Suppression\_Dépot
      * Nouveau\_Forfait, MAJ\_Forfait, ~~Suppression\_Forfait~~
      * Nouvelle Notoriété, Maj\_Notoriété, Suppression…
      * … Ville
      * … Pays
    - Méthode « Sauvegarde/Confirmation » qui appelle la DAL
* **Erreurs**
* **WebApi**
  + Méthodes traitant les requêtes Put ; Get etc..

***Lien entre les couches***

1. Création/Maj des entités en Disconnected state via les méthodes de la classe BL
2. Sauvegarde (SaveChange) des ajouts/Changements dans la DB via les méthodes de la classe DAL. Les méthodes de la classe DAL prendront en paramètre les entités créées/Mise à jour par la classe BL

Les modèles sont mis à jour automatiquement par Entity Framework.

## Partie MVC