BTS SERVICES INFORMATIQUES AUX ORGANISATIONS

E5 : Production et fourniture de services informatiques

SESSION 2021

Durée : 4h00 Coefficient : 5

Cas Restiloc

Ce sujet comporte 17 pages dont 10 pages de documentation.

La candidate ou le candidat doit vérifier que le sujet qui lui a été remis est complet.

Aucun matériel ni document autorisé

Barème

Mission 1	Gestion des prestations de remise en état (PREE)	20 points
Mission 2	Gestion des rendez-vous	30 points
Mission 3	Application mobile pour les experts	40 points
Mission 4	Gestion de projet	10 points
	Total	100 points

Dossier documentaire

Présentation du contexte

De plus en plus, particuliers et professionnels choisissent pour leur(s) véhicule(s) de signer un contrat de location avec une société de financement sur une période qui est le plus souvent fixée entre 12 et 48 mois et peut aller jusqu'à 60 mois. Ces contrats sont signés par le biais de garages liés à une ou plusieurs marques automobiles.
À la fin du contrat, si le véhicule est restitué, il doit être expertisé afin de relever tous les dommages qu'il a subis durant la période de location. Cette expertise peut conduire à des frais de remise en état qui seront facturés au client par la société de financement.

L'organisation cliente
La société RestiLoc, implantée dans tous les pays d'Europe, est spécialisée dans l'expertise de véhicules en fin de contrat, pour le compte des sociétés de financement.
Une bonne couverture nationale de son réseau d'experts est indispensable à RestiLoc pour remplir ses missions d'expertise en maintenant un haut niveau de service. La filiale RestiLoc France compte ainsi environ 100 experts, répartis sur tout le territoire français.

Le prestataire de services informatiques
La direction des systèmes d'information (DSI) de RestiLoc, située sur le territoire français, est chargée d'assurer la mise en place et la maintenance des infrastructures réseau ainsi que de développer les logiciels métiers nécessaires à l'organisation des expertises. Ainsi la DSI a développé le logiciel Torapix permettant notamment aux experts d'accéder :

- à la planification de leurs missions d'expertise ;
- aux informations de chaque mission : société de financement concernée, client qui effectue la location et détails sur le véhicule (marque, modèle, finition, puissance, etc.).

Le logiciel Torapix s'appuie sur une base de données *SQL Server* hébergée par la DSI de RestiLoc.

En tant qu'employé(e) du service informatique de RestiLoc, vous intervenez sur plusieurs projets afin de répondre aux demandes d'évolutions du logiciel Torapix, sous la responsabilité directe de M. Poutia, directeur de la DSI.

Mission 1 – Gestion des prestations de remise en état (PREE)

Document à utiliser : 1

La société RestiLoc expertise les véhicules restitués en fin de contrat de location, pour le compte des sociétés de financement. Dans ce cadre, le rôle de l'expert est d'effectuer un état des lieux de chaque véhicule restitué de façon à informer précisément la société de financement sur les besoins de remise en état. À partir de cette expertise et de son propre barème de dépréciation, la société de financement facturera le client et décidera d'effectuer ou pas, des travaux de remise en état avant de mettre ou de faire mettre le véhicule sur le marché de l'occasion. RestiLoc ne s'occupe donc ni de chiffrer en euros les prestations de remise en état nécessaires, ni de réparer ou de faire réparer.
Dans le cadre de l'évolution du système d'information de RestiLoc, il est demandé à la DSI de faire évoluer l'application Torapix de manière à renseigner le détail des prestations de remise en état nécessaires (PREE) identifiées par l'expert pour chaque dossier.

Un dossier d'expertise correspond à un seul véhicule. L'expert est chargé de relever les éventuelles PREE nécessaires. Ces prestations seront identifiées par la référence du dossier d'expertise suivie d'un numéro d'ordre (numéro séquentiel de 1 à x). Chaque prestation sera décrite par un libellé (texte court) et pourra être précisée par une description (texte long). Au cours de cette expertise, une photo est réalisée pour preuve, pour chaque prestation de remise en état nécessaire. Le nom de cette photo est à mémoriser.
Il existe deux types de prestations de remise en état :

- les prestations "peinture" ;
- les prestations "pièce".

Une prestation de **remise en état "peinture"** concerne des travaux sur un ou plusieurs éléments de carrosserie du véhicule restitué (par exemple : pare-chocs avant, aile avant gauche, …), chacun nécessitant un traitement de peinture (par exemple : "réaliser un voile", "réparer et peindre partiellement" ou "réparer et peindre complètement", …).

Une prestation de **remise en état "pièce"** concerne le remplacement de pièces lorsque celles-ci sont cassées ou très abimées Il faut mémoriser quelles sont les pièces à changer (par exemple : boite à gants, essuie-glace, roue de secours, …) et dans quelle quantité, sachant que chaque pièce possède une référence et un libellé.

Question 1.1
Proposer une modification de la structure de la base de données pour intégrer la gestion des PREE.

IMPORTANT : la candidate ou le candidat présentera les évolutions de la structure de la base de données en adoptant le formalisme de son choix (schéma entité-association, diagramme de classes ou encore schéma relationnel).

Mission 2 – Gestion des rendez-vous

Documents à utiliser : 2, 3 et 4

IMPORTANT : la candidate ou le candidat peut choisir de présenter les éléments de code à l'aide du langage de programmation de son choix ou de pseudo-code algorithmique.

Lorsqu'une société de financement missionne RestiLoc afin d'effectuer une expertise, elle lui indique le type de rendez-vous et toutes les données nécessaires.
Un des employés du service planification de RestiLoc est chargé de contacter les clients, ou les garages, afin de planifier les rendez-vous des experts.

Il existe deux types de rendez-vous :

- le rendez-vous client : l'expert rencontre directement le loueur du véhicule, effectue l'expertise en sa présence, puis lui fait signer le compte rendu de restitution ;
- le *pool* garage : le client dépose son véhicule dans un garage, avant que l'expert ne soit missionné. Une fois l'expertise effectuée, le compte rendu de restitution sera signé et tamponné par un employé du garage.

Afin de gérer ces rendez-vous, une application écrite en C# a commencé à être développée par la DSI. M. Poutia vous demande de la compléter afin qu'elle réponde mieux aux besoins, notamment qu'elle permette une meilleure analyse des rendez-vous qui n'ont pas abouti. En effet, lorsqu'un expert se présente à un rendez-vous et qu'il ne peut pas expertiser le véhicule, on parle d'indisponibilité. Dans ce cas l'expert crée une indisponibilité en renseignant le motif et indique également si le client est responsable ou non de cette indisponibilité. Les motifs sont toujours les mêmes : "Client absent", "Véhicule inaccessible", "Véhicule absent", "Adresse erronée".

Question 2.1
Justifier le choix fait par vos collègues d'utiliser une classe abstraite pour la classe Expertise.

Question 2.2
Écrire la méthode *AjouterExpertisePool* de la classe SocieteFinancement qui permet d'ajouter une nouvelle expertise de type Pool_Garage à la collection des expertises d'une société de financement.

Le responsable du service planification constate une augmentation des indisponibilités. Afin d'optimiser la gestion des rendez-vous, il souhaiterait obtenir différentes statistiques sur les motifs d'annulation des rendez-vous. Il a donc demandé à la DSI d'ajouter des traitements qui lui permettront d'analyser les situations.

Question 2.3
Écrire la méthode *GetMotif* de la classe Indisponibilite qui retourne le motif de l'indisponibilité.

Question 2.4
Écrire la méthode *LesExpertisesIndispos* de la classe SocieteFinancement qui retourne la collection de toutes les expertises qui ont donné lieu à une indisponibilité.

Question 2.5
Écrire la méthode *NbIndisponibilites* qui retourne le nombre d'expertises indisponibles correspondant au motif passé en paramètre.

Mission 3 – Application mobile pour les experts

Documents à utiliser : 5, 6, 7, 8 et 9

Pour permettre aux experts de réaliser leurs tournées en visitant les garages dans lesquels se trouvent les véhicules à expertiser, et ceci dans les meilleures conditions possibles, il a été décidé de développer une application mobile qui leur permettra de visualiser leurs missions d'expertises du jour. Dans cette première phase du projet, seule la version pour les équipements *Android* a été développée en langage *Java* en utilisant l'outil *Android Studio*.
Cette application s'appuiera sur les données mises à disposition via des services *Web*.

Dans un premier temps, vous avez en charge la réalisation de diverses requêtes concernant les véhicules à expertiser ainsi que l'activité des garages partenaires et des experts :

1. Liste des informations (date, heure, immatriculation, marque, modèle, nom et prénom de l'expert) des véhicules à expertiser, à trier par date et heure croissantes ;
2. Nombre de véhicules expertisés en 2018 pour chaque expert (nom et prénom) ;
3. Liste des garages (identifiant, nom, ville et téléphone) dans lesquels il y a eu plus de 100 missions créées.

Question 3.1
Écrire en langage *SQL* les requêtes permettant d'obtenir les informations souhaitées.

Une *API* (*Application Programming Interface* ou interface de programmation applicative) est développée afin de faciliter l'appel des services *Web* qui se fera le plus simplement possible par une requête *HTTP*, par exemple :

http://www.restiloc.fr/api/getLesMissionsByIdExpert.php?idExpert=2

Le paramètre est passé par la méthode GET.

Le code du service *Web getLesMissionsByIdExpert* permet de retourner à l'application mobile les missions correspondant à la tournée du jour pour l'expert dont l'identifiant est passé en paramètre, et ceci au format *JSON* (*JavaScript Object Notation*).

M. Poutia vous demande de documenter l'architecture applicative pour expliquer où va s'exécuter chaque code (*Web service getLesMissionsByIdExpert*, code des classes GestionURL, MainActivity et ListAdapter et les requêtes SQL) et quels sont les échanges entre les différents éléments de l'application :

- le SGBD *SQLServer* ;
- le serveur HTTP *Apache* ;
- l'application *Android* implantée dans l'ordiphone *(smartphone).*

Question 3.2
Proposer un schéma de votre choix qui répond à la demande de M. Poutia.

Question 3.3
Dans une courte note (environ 10 lignes) :

- Vous exposerez votre diagnostic <u>en terme de sécurité</u> à propos de l'appel des services web par API, tel qu'il est présenté ci-dessus,
- Vous présenterez ensuite vos préconisations, en décrivant les différents aspects sécuritaires que vous voulez corriger.

M. Poutia vous fait remarquer que l'interface graphique obtenue résultant du code actuel n'est pas conforme à la maquette souhaitée : le nom du garage n'apparait pas. Il vous demande de corriger cette erreur.

Question 3.4
Corriger cette erreur en intervenant là où il est nécessaire de modifier le code.

Sur votre copie vous indiquerez les fichiers et les numéros des lignes concernées.

Question 3.5
Après avoir rappelé la fonction et de quoi est fait un test unitaire de manière classique, vous préciserez en quoi des tests unitaires auraient ou non permis de déceler l'erreur corrigée à la question précédente

Mission 4 – Gestion de projet

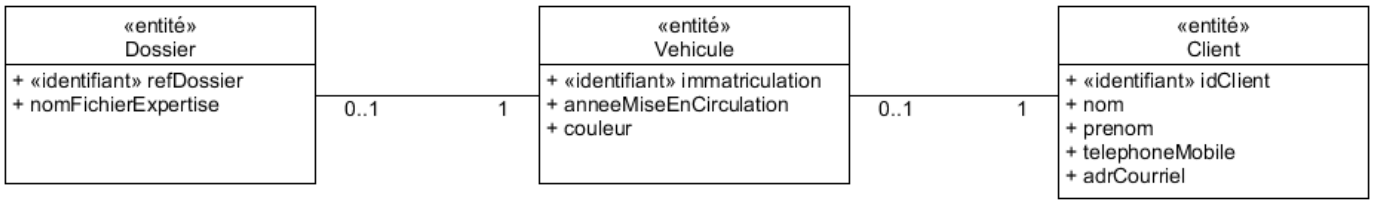
Document à utiliser : 10

M POUTIA a reçu une société spécialisée en sécurité, qui a audité l'entreprise. L'objectif est d'obtenir la certification ISO 27001.
Après quelques semaines, le rapport de sécurité arrive sur le bureau de M. POUTIA. Celui-ci vous convoque alors pour dépouiller avec lui les réponses et recommandations de la société.

N° défaillance	Détails de la défaillance	Périmètre concerné	Niveau de risque
001	L'outil de certification met en évidence un risque d'injection SQL sur le service getLesMissionsByIdExpert.php.	Application Torapix	élevé
002	L'outil de certification met en évidence un risque d'injection SQL sur le service ...	Application Torapix	élevé
003	…	Application Torapix	élevé
004	Non-respect du RGPD sur l'application Torapix les véhicules et les clients	Application Torapix	faible
005	Risque de phishing et ransomware élevé par mauvais usage des utilisateurs du système d'information	Entreprise	moyen

Question 4.1 (3 points)
Expliquez ce qu'est une injection SQL, en précisant quelle partie du code nécessite une correction
Proposez cette correction en vous aidant des fonctions fournies dans les documents

Pour la défaillance N°4, M POUTIA suspecte la modification récente de l'application permettant d'indiquer au client quand venir chercher son véhicule par SMS ou par messagerie électronique.



Question 4.2 (3 points)
Dans une courte note, décrivez ce que le RGPD implique et en quoi la défaillance N°4 s'applique

Enfin, M POUTIA vous demande de faire des propositions pour lutter contre les risques de la défaillance N°5

Question 4.3 (4 points)
a) Expliquer rapidement les termes phishing et ransomware (1 à 2 lignes pour chaque terme).
b) Proposez une ou deux solutions pour améliorer les usages et inciter les utilisateurs à ne pas commettre de fautes.

DOCUMENT 1 : Extrait du schéma de données de la base de données existante

Schéma conceptuel des données

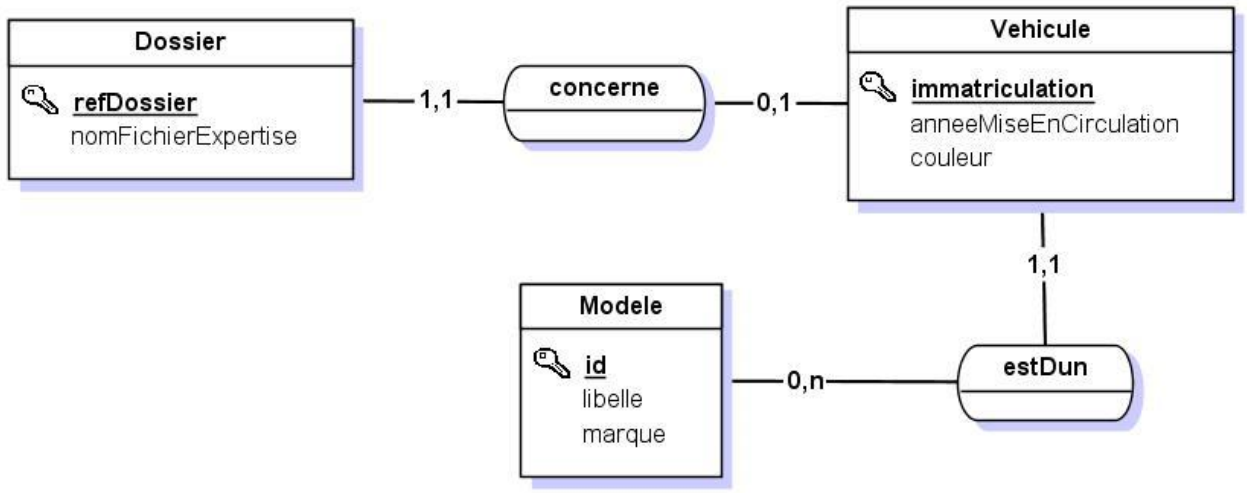
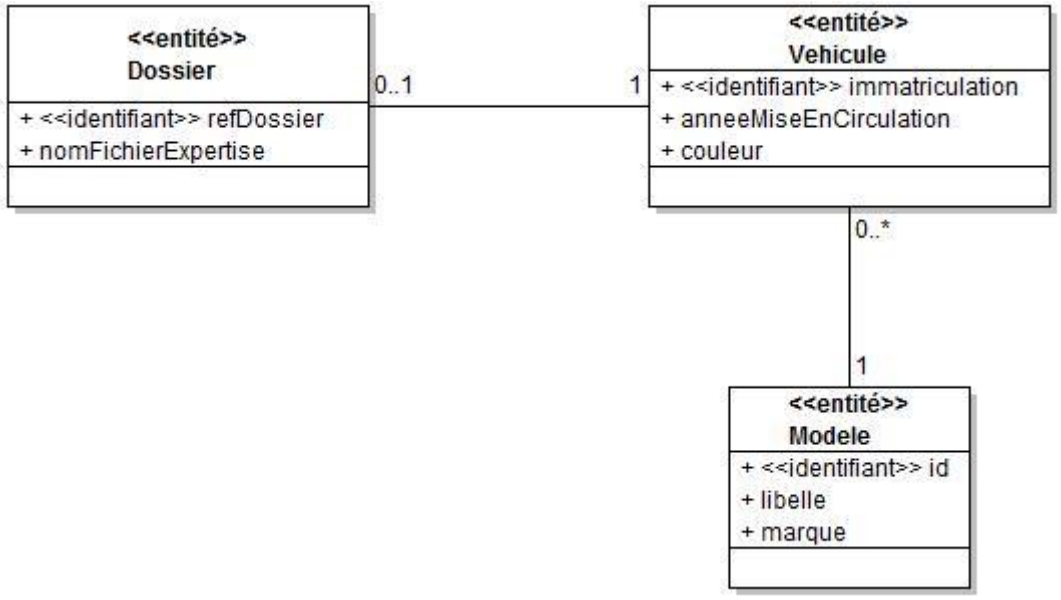


Diagramme de classes



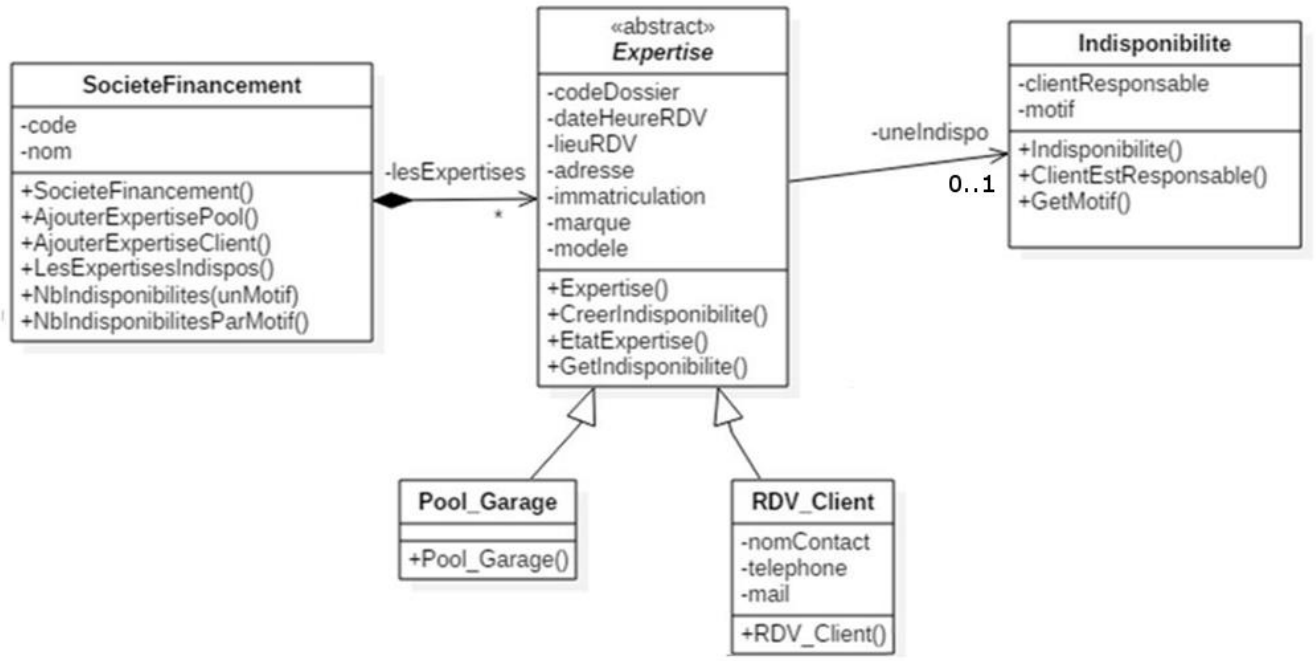
Modèle relationnel

Dossier (refDossier, nomFichierExpertise, immatriculation)
Clé primaire : refDossier
Clé étrangère :
immatriculation en référence à immatriculation de la table Vehicule

Vehicule (immatriculation, anneeMiseEnCirculation, couleur, idModele, refDossier)
Clé primaire : immatriculation
Clés étrangères :
idModele en référence à id de la table Modele
refDossier en référence à refDossier de la table Dossier (cet attribut est à *null* tant qu'il n'y a pas eu de dossier d'expertise)

Modele (id, libelle, marque)
Clé primaire : id

DOCUMENT 2 : Diagramme de classes correspondant à la gestion des rendez-vous



DOCUMENT 3 : Description textuelle des classes métier

```
public class SocieteFinancement
{
   private string code;
   private string nom;
   private List<Expertise> lesExpertises;

   // Constructeur qui initialise les attributs privés et instancie la collection lesExpertises
   public SocieteFinancement (string codeSL, string nomSL) { … }

     public void AjouterExpertiseClient (string dossier, DateTime dateHeure, string lieu, string adresse,
          string immat, string marque, string modele, string unContact, string unTel, string unMail) { … }

   // Méthode qui permet d'ajouter une nouvelle expertise de type pool garage
   public void AjouterExpertisePool (string dossier, DateTime dateHeure, string lieu,
                                     string adresse, string immat, string marque, string modele)
   {
                  // À compléter sur votre copie
   }

   // Méthode qui retourne la liste des expertises indisponibles
   public List<Expertise> LesExpertisesIndispos ()
   {
                  // À compléter sur votre copie
   }

   // Méthode qui retourne le nombre d'expertises indisponibles correspondant au motif leMotif
   public int NbIndisponibilites(string leMotif)
   {
                  // À compléter sur votre copie
   }
}
public abstract class Expertise
{
   private string codeDossier ;
```

```
    private DateTime dateHeureRDV;
    private string lieuRDV ;
    private string adresse ;
    private string immatriculation
    private string marque ;
    private string modele ;
    private Indisponibilite uneIndispo;

    //Constructeur permettant de valoriser les attributs d'un objet
    public Expertise(string codeDossier, DateTime uneDateHeure, string unLieu,
          string uneAdresse, string uneImmat, string uneMarque, string unModele)
    {  …  }

    //Méthode qui détermine l'état de l'expertise sous la forme d'un entier
    //<returns>1 – à réaliser ; 2 – réalisée ; 3 - indisponible</returns>
    public int EtatExpertise() { ... }

    //Méthode qui permet de créer l'indisponibilité de l'expertise qui n'a pas pu avoir lieu
    public void CreerIndisponibilite(string unMotif, bool clientResponsable) {...}


    //Obtient l'objet de type Indisponibilite si l'expertise n'a pas pu avoir lieu
    //<returns>objet de type Indisponibilite ou valeur null si expertise à réaliser ou réalisée</returns>
    public Indisponibilite GetIndisponibilite() { ... }
}
```

```
public class RDV_Client : Expertise {          //RDV_Client hérite de la classe Expertise
    private string nomContact;
    private string telephone;
    private string mail;


    //Constructeur qui permet d'instancier une expertise de type RDV client
    public RDV_Client (string codeDossier, DateTime uneDateHeure, string unLieu,
          string uneAdresse, string uneImmat, string uneMarque, string unModele, string contact,
          string tel, string mail) : base(codeDossier, uneDateHeure, unlieu, uneAdresse, uneImmat,
          uneMarque, un Modele) { … }
}
```

```
public class Pool_Garage : Expertise {        //Pool_Garage hérite de la classe Expertise

    //Constructeur qui permet d'instancier une expertise de type Pool garage
    public Pool_Garage (string codeDossier, DateTime uneDateHeure, string unLieu,
          string uneAdresse, string uneImmat, string uneMarque, string unModele) : base(codeDossier,
          uneDateHeure, unlieu, uneAdresse, uneImmat, uneMarque, un Modele) { … }
}
```

```
public class Indisponibilite
{

   private string motif;
   private bool clientResponsable;

   //Constructeur qui permet d'instancier une indisponibilité de RDV
   public Indisponibilite(string unMotif, bool clientResponsable) { … }

   // indique si le client est responsable ou non de l'indisponibilité
   //<returns>True : client responsable / False : client non responsable</returns>
   public bool ClientEstResponsable() { ... }

   //Obtient le motif de l'indisponibilité
   public string GetMotif()
   {

         // À compléter sur votre copie


   }
}
```

DOCUMENT 4 : Exemple d'utilisation d'une collection

```
List<Indisponibilite> lesIndispos = new List<Indisponibilite>() ; // Déclaration et instanciation d'une collection d'instances d'Indisponibilite
Indisponibilite uneIndispo = new Indisponibilite(…) ;
lesIndispos.Add(uneIndispo) ; // Ajout d'un élément à la collection lesIndispos[0] ; // Obtient le premier élément de la collection lesIndispos.Count ; // Obtient le nombre d'éléments présents dans la collection foreach(Indisponibilite indispo in lesIndispos) {

      // Parcours de tous les éléments de la collection
      // indispo étant l'objet en cours dans le parcours
}
```

DOCUMENT 5 : Extrait du schéma relationnel concerné par les services *Web*

Expert (idExpert, nom, prenom, email, login, mdp)
Clé primaire : idExpert

Garage (idGarage, nom, adnum, rue, cp, ville, tel, latitude, longitude)
Clé primaire : idGarage

MissionExpertise (idMission, dateMission, heureDebut, idGarage, idExpert, kilometresCompteur, immatriculation)
Clé primaire : idMission
Clés étrangères :
idExpert fait référence à la clé primaire idExpert de la table Expert
idGarage fait référence à la clé primaire idGarage de la table Garage

Vue VehiculeExpertise (immatriculation, marque, modele, anneeMiseEnCirculation, couleur, idMission)

Remarque : **VehiculeExpertise** est une vue SQL qui agrège les informations nécessaires aux experts concernant uniquement les véhicules à expertiser.

DOCUMENT 6 : Code PHP du service *Web getLesMissionsByIdExpert.php*

```
1.  <?php
2.    try {
3.          // Connexion à la base de données SQLServer
4.          $bdd = new PDO('sqlsrv:server=…;Database=…,'…','…');

5.          // Récupération de l'idExpert
6.          $id = $_GET['idExpert'];

7.          // Création de la requête SQL
8.  $sql = "SELECT idMission, heureDebut, ville, V.immatriculation AS immatriculation, marque,
    modele";
9.          $sql = $sql." FROM MissionExpertise M";
10.         $sql = $sql." JOIN Garage G ON M.idGarage = G.idGarage";
11.         $sql = $sql." JOIN VehiculeExpertise V ON M.idMission = V.idMission";
12.         $sql = $sql." WHERE idExpert = ".$id;
13.         $sql = $sql." AND dateMission = '".DATE("Y-m-d")."'";
14.         $sql = $sql." ORDER BY heureDebut ASC";
15.         // La fonction DATE(…) permet de récupérer la date du jour au format donné

16.         // Exécution de la requête SQL
17.         $reponse = $bdd->query($sql);

18.         // Conversion de la réponse sous la forme d'un tableau associatif
19.         $resultat = $reponse->fetchAll(PDO::FETCH_ASSOC);
20.   }
21.   catch (Exception $e) {
22.         die('Erreur : ' . $e->getMessage());
23.   }
24.   // Encodage du résultat au format JSON
25.   echo(json_encode($resultat));
26. ?>
```

DOCUMENT 7 : Exemple de résultat au format *JSON* retourné par le service *Web getLesMissionsByIdExpert*

[{"idMission":2,"heureDebut":"10:30","ville":"Montpellier","immatriculation":"AA-229-XY","marque":"Citroën","modele":"C3"},

{"idMission":7,"heureDebut":"11:30","ville":"Lavérune","immatriculation":"BC-548-ZX","marque":"Peugeot","modele":"208"},
{"idMission":6,"heureDebut":"15:20","ville":"Béziers","immatriculation":"AB-346-TU","marque":"Renault","modele":"Clio"}]

DOCUMENT 8 : Code du fichier *list_layout.xml* de l'application mobile et maquette de l'interface graphique souhaitée

Ce code de l'interface graphique définit une ligne de l'écran d'affichage des missions du jour sur le *smartphone*.

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.  xmlns:tools="http://schemas.android.com/tools"
4.  android:layout_width="match_parent"
5.  android:layout_height="match_parent"
6.  android:padding="10dp" >

7.      <TextView
8.      android:id="@+id/idM"
9.      android:layout_width="wrap_content"
10.     android:layout_height="wrap_content"
11.     android:layout_marginLeft="10dp"
12.     android:textStyle="bold"
13.     tools:text="Id" />

14.     <TextView
15.     android:id="@+id/heureM"
16.     android:layout_width="wrap_content"
17.     android:layout_height="wrap_content"
18.     android:layout_toRightOf="@id/idM"
19.     android:layout_marginLeft="10dp"
20.     android:textStyle="bold"
21.     tools:text="Heure" />

22.     <TextView
23.     android:id="@+id/villeM"
24.     android:layout_width="wrap_content"
25.     android:layout_height="wrap_content"
26.     android:layout_toRightOf="@id/heureM"
27.     android:layout_marginLeft="10dp"
28.     tools:text="Ville" />

29.     <TextView
30.     android:id="@+id/immatM"
31.     android:layout_width="wrap_content"
32.     android:layout_height="wrap_content"
33.     android:layout_marginLeft="10dp"
34.     android:layout_below="@id/heureM"
35.     tools:text="Immatriculation" />

36.     <TextView
37.     android:id="@+id/marqueModeleM"
38.     android:layout_width="wrap_content"
39.     android:layout_height="wrap_content"
40.     android:layout_toRightOf ="@id/immatM"
41.     android:layout_below="@id/villeM"
42.     android:layout_marginLeft="10dp"
43.     tools:text="Marque Modèle" />
44. </RelativeLayout>
```

Maquette de l'interface graphique souhaitée

DOCUMENT 9 : Codes *Android Java* de l'application mobile

Code du fichier *GestionURL.java* proposant les méthodes pour se connecter au service *Web* et renvoyer un résultat au format *JSON*

```
1.  import …;

2.  public class GestionURL {
3.      // Méthode qui soumet la requête http et qui retourne le fichier JSON
4.      public static String get(String url) throws IOException {
5.          InputStream is = null;
6.          try {
7.              final HttpURLConnection conn = (HttpURLConnection) new
8.                                      URL(url).openConnection();
9.              // Utilisation de la méthode GET
10.             conn.setRequestMethod("GET");
11.             conn.setDoInput(true);

12.             // Début de la requête http
13.             conn.connect();
14.             is = conn.getInputStream();

15.             // Conversion du résultat au format String
16.             return convertToString(is);
17.         }
18.         catch (Exception e){
19.             Log.e("Erreur de connexion",e.toString());
20.         }
21.         finally {
22.             // Fermeture du flux
23.             if (is != null) {
24.                 is.close();
25.             }
26.         }
27.     }

28.     // Méthode qui convertit le flux en entrée au format String
29.     private static String convertToString(InputStream is) throws IOException { … }
30. }
```

Code du fichier *MainActivity.java* qui va récupérer le fichier *JSON* et lancer l'affichage de la listView

```
1.  import …

2.  public class MainActivity extends AppCompatActivity {
3.      protected void onCreate(Bundle savedInstanceState) {
4.          super.onCreate(savedInstanceState);
5.          setContentView(R.layout.activity_main);

6.          String fichierJSON = null;
            try {
7.              String url = "http://www.restiloc.fr/api/getLesMissionsByIdExpert.php?idExpert=2";

8.              // Récupération du fichier JSON
9.              fichierJSON = GestionURL.get(url)

10.             // transformation du fichier JSON en tableau JSON
11.             JSONArray ja = new JSONArray(fichierJSON);

12.             // Remplissage d'une ArrayList avec le tableau JSON
13.             ArrayList<JSONObject> listItems=getArrayListFromJSONArray(ja);
14.             ListView listV=(ListView)findViewById(R.id.listv);

15.             // Affichage du titre
16.             SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
17.             String dateDuJour = sdf.format(new Date());
18.             TextView titre = new TextView(this);
19.             titre.setText("Liste des missions du " + dateDuJour);
20.             listV.addHeaderView(titre);

21.             // Instanciation de la ListAdapter
22.             ListAdapter adapter = new ListAdapter(this,R.layout.list_layout,R.id.idE,listItems);

23.             // Affichage de la ListView dans l'interface graphique
24.             listV.setAdapter(adapter);

25.         } catch (IOException e) {
26.             Log.e("On Create Erreur IO",e.toString());
27.         } catch (JSONException e) {
28.             Log.e("On Create Erreur Json",e.toString());
29.         }
30.     }

31.     // Méthode qui remplit une ArrayList à partir d'un tableau JSON
32.     private ArrayList<JSONObject> getArrayListFromJSONArray(JSONArray jsonArray){ … }
33. }
```

Par souci de simplification, l'identifiant de l'expert a été positionné arbitrairement à la valeur 2, dans l'application finale il sera déterminé après authentification de l'expert

Code du fichier *ListAdapter.java* qui a pour rôle de retourner la vue des missions avec les valeurs contenues dans le fichier *JSON*

```
1.  import …

2.  public class ListAdapter extends ArrayAdapter<JSONObject>{
3.      private int vg;
4.      private ArrayList<JSONObject> list;
5.      private Context context;

6.      public ListAdapter(Context context, int vg, int id, ArrayList<JSONObject> list){
7.          super(context,vg, id,list);
8.          this.context = context;
9.          this.vg=vg;
10.         this.list=list;
11.     }

12.     // méthode appelée automatiquement lors de l'affichage de la liste
13.     public View getView(int position, View convertView, ViewGroup parent) {
14. LayoutInflater inflater = (LayoutInflater)
    context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
15.         View itemView = inflater.inflate(vg, parent, false);

16.         // Instanciation des TextView rattachées à l'interface graphique
17.         TextView txtId=(TextView)itemView.findViewById(R.id.idM);
18.         TextView txtHeure=(TextView)itemView.findViewById(R.id.heureM);
19.         TextView txtVille=(TextView)itemView.findViewById(R.id.villeM);
20.         TextView txtImmat=(TextView)itemView.findViewById(R.id.immatM);
21.         TextView txtMarqueModele=(TextView)itemView.findViewById(R.id.marqueModeleM);

22.         try {
23.             // Renseignement des TextView
24.             txtId.setText(list.get(position).getString("idMission"));
25.             txtHeure.setText(list.get(position).getString("heureDebut"));
26.             txtVille.setText(list.get(position).getString("ville"));
27.             txtImmat.setText(list.get(position).getString("immatriculation"));
28.             txtMarqueModele.setText(list.get(position).getString("marque") + " " +
29.                                     list.get(position).getString("modele"));
30.         }
31.         catch (JSONException e) {
32.             Log.e("Erreur getView",e.toString());
33.         }
34.         return itemView;
35.     }
36. }
```

DOCUMENT 10 : Fonctions sur les chaînes de caractères

htmlentities

htmlentities — Convertit tous les caractères éligibles en entités HTML
htmlentities() est identique à la fonction htmlspecialchars(), sauf que tous les caractères qui ont des équivalents en entités HTML sont effectivement traduits.
ENT_QUOTES Convertit les guillemets doubles et les guillemets simples.

```
echo htmlentities($str, ENT_QUOTES, "UTF-8");
```

htmlspecialchars

htmlspecialchars — Convertit les caractères spéciaux en entités HTML
ENT_QUOTES Convertit les guillemets doubles et les guillemets simples.

```
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
```

mysqli_real_escape_string

mysqli::real_escape_string -- mysqli_real_escape_string — Protège les caractères spéciaux d'une chaîne pour l'utiliser dans une requête SQL, en prenant en compte le jeu de caractères courant de la connexion

Les caractères encodés sont NUL (ASCII 0), \n, \r, \, ', ", and Control-Z.

```
$mysqli->real_escape_string($city));
```

DOCUMENT 11 : Fonctions sur les requêtes SQL

Exemple de requête préparée

```
<?php
// séquence de connexion
$stmt = $dbh->prepare("INSERT INTO REGISTRY (nom, valeur) VALUES (:nom, :valeur)");
$stmt->bindParam(':nom', $nom); $stmt->bindParam(':valeur', $valeur);
$valeurs = range('a', 'z');
foreach($valeurs as $valeur => $nom) {
	// insertion d'une ligne
	$stmt->execute();
}
// séquence de déconnexion
?>
```

kklk