|  |  |
| --- | --- |
| **BTS SERVICES INFORMATIQUES AUX ORGANISATIONS** | **SESSION 2014** |

E5SL : PRODUCTION ET FOURNITURE DE SERVICES

|  |  |
| --- | --- |
| **Durée : 4 heures** | **Coefficient : 5** |

CAS AÉROPLAN

Ce sujet comporte 17 pages dont 10 pages de documentation.

La candidate ou le candidat est invité-e à vérifier qu’il-elle est en possession d’un sujet complet.

Aucun matériel ni document autorisé

**Dossier documentaire :**

*Document 1 – Extrait du diagramme de classes de l'application*

*Document 2 – Description des classes métier (extrait)*

*Document 3 – Exemple d'utilisation d'une collection*

*Document 4 – Ajout de tests unitaires pour la classe "Mouvement"*

*Document 5 – Fiche d'incident n°D24*

*Document 6 – Évolution des classes "Mouvement" et "Retard"*

*Document 7 – Présentation de la base de données*

*Document 8 – Documents concernant les équipages*

*Document 9 – Fonctionnalités du SGBD*

*Document 10 – Fonction stockée "get\_indemnisation\_globale"*

## Barème

|  |  |  |
| --- | --- | --- |
| Mission 1 | Corriger l'application de gestion des retards | 20 points |
| Mission 2 | Négocier avec les aéroports | 30 points |
| Mission 3 | Gérer les équipages | 20 points |
| Mission 4 | Étude prévisionnelle de l'indemnisation des retards | 30 points |
|  | Total | 100 points |

**Présentation du contexte**

*Avionix* est une entreprise de services du numérique (ESN) spécialisée dans l'avionique, créée en 1992 et située dans le sud francilien.

Créée en 1992, sa principale activité est le développement de progiciels, en particulier dans les domaines suivants :

* gestion de compagnies aériennes,
* gestion des échanges de données entre les avions et les systèmes d’information au sol.

Plus de cinquante compagnies aériennes, réparties sur presque tous les continents, utilisent les logiciels développés par cette société. En 2011, *Avionix* a réalisé plus de 65% de son chiffre d’affaires à l’exportation.

Son personnel est composé de spécialistes de l'aviation et de techniciens informatiques.

*Avionix* a réussi à faire sa place dans son créneau principalement grâce à un client fidèle : la compagnie aérienne *Aéroplan*, pour laquelle elle a conçu des applications ciblées (planification des vols, gestion de flotte, ...).

*Aéroplan* créée en 1981, principal client d’*Avionix*, couvre 14 destinations. Grâce à une flotte de 7 appareils, elle transporte près de 1 300 000 passagers par an.

Dans un souci permanent d’amélioration des services aux passagers, *Aéroplan* s'est engagée dans une démarche globale de qualité. Dans ce but, elle utilise plusieurs applications d'*Avionix* dont *TripReport*, codée en Java. Cette application améliore la gestion des vols en offrant au pilote la possibilité de saisir les informations concernant le vol depuis un appareil mobile. Ces informations sont ensuite centralisées dans la base de données générale d'*Aéroplan*.

Les informations traitées par *TripReport* sont pour l'instant assez limitées et *Aéroplan* souhaite que l'application offre plus de possibilités. Elle a donc demandé des améliorations à *Avionix*. Employé-e au service développement d'applications chez *Avionix*, vous faites partie de l'équipe chargée de répondre aux besoins d’*Aéroplan*.

**Mission 1 – Corriger l'application de gestion des retards**

Documents à utiliser : 1, 2, 3, 4 et 5

*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.*

Lors d’un vol, le commandant de bord (pilote de l’avion) doit rendre compte des principaux événements de vol (parking, décollage, trajet, atterrissage, ...). Cela se faisait auparavant via un rapport de vol manuscrit qui nécessitait une saisie ultérieure. Ce type de compte-rendu posait plusieurs problèmes :

* le suivi des événements de vol ne pouvait être fait rapidement,
* la saisie des rapports était longue et source d’erreurs.

Rapidement ces rapports ont été saisis directement par le pilote via un appareil nomade grâce à l'application *TripReport*. Aujourd’hui les compagnies aériennes privilégient les tablettes (plus pratiques, plus grandes, ...) aux dépens des *smartphones*.

L'application *TripReport*, permet ainsi de saisir des informations générales de vol telles que :

* le code et numéro de vol,
* les heures de départ et d'arrivée,
* les aéroports de départ et d'arrivée,
* l'avion utilisé.

Afin d’améliorer le fonctionnement et donc le service rendu aux passagers, il est important de connaître les retards, les incidents de vol, ou encore les contrôles effectués au départ et à l’arrivée. Les rapports numériques sont pour l'instant très succincts et nécessitent de conserver des documents manuscrits, lourds à gérer.

**D**ans un premier temps, les améliorations portent sur l'enregistrement des retards. Un mouvement correspond à un vol pour une date précise, à des heures de départ et d'arrivée prévues. Tout retard touchant un mouvement doit être conservé. . Le pilote devra donc pouvoir saisir les retards ayant affecté le vol dont il a la charge. Pour cela, les membres de votre équipe ont déjà complété la classe "Mouvement" et ajouté les classes "Retard" et "TypeRetard" aux classes métier déjà présentes.

**D**es tests unitaires ont été ajoutés à la classe de test de la classe "Mouvement" pour prendre en compte les nouvelles méthodes. Cependant, les tests réalisés n'ont pas permis de détecter tous les problèmes. Un utilisateur vous envoie la fiche d’incident n°D24.

Vous êtes chargé-e de poursuivre le travail déjà débuté et de mettre à jour la documentation technique.

## Votre mission

**1.1 À partir de la description des classes métier, indiquer sur votre copie les compléments au diagramme de classes nécessaires pour prendre en compte la gestion des retards.**

**1.2 Rédiger la rubrique "Description de la cause" de la fiche d’incident D24.**

**1.3 Proposer le code permettant de corriger cette erreur.**

**1.4 Rédiger le test unitaire qui aurait permis de détecter cette erreur.**

**Mission 2 – Négocier avec les aéroports**

Documents à utiliser : 2, 3 et 6

*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.*

Les négociations entre les compagnies aériennes et les aéroports sont complexes et stratégiques. Les sommes en jeu sont très importantes. Par exemple, lors d’un retard, il est primordial d’identifier le responsable pour gérer les indemnisations.

Il est décidé de profiter de l’application *TripReport* pour recueillir directement auprès du pilote des informations utiles sur la responsabilité des immobilisations des avions.

On demandera donc au pilote du vol de noter sur sa tablette s’il pense que la responsabilité d’un retard incombe à l’aéroport.

**V**otre chef de projet commente l'évolution prévue des classes "Mouvement" et "Retard" présentée dans le dossier documentaire :

* Dans la classe Retard, il faut conserver l’attribut "commentaire" librement saisie par le pilote.
* On ajoute l’attribut "impliqueAeroport" qui précise si, aux yeux du pilote, l’aéroport a une responsabilité dans le retard, la méthode "getImpliqueAeroport" permettra d'y accéder.
* Dans la classe TypeRetard, on conserve l’attribut "codeSituation" qui permet d’identifier l’aéroport impliqué : la valeur "AV" (avant vol) pour l’aéroport de départ et la valeur "AP" (après vol) pour l’aéroport d’arrivée. La situation de valeur "EV" (en vol) ne concerne pas les aéroports.

**L**ors de l’immobilisation d’un avion sur un aéroport (au décollage ou à l'atterrissage), l’intervention du pilote est souvent déterminante pour obtenir une solution rapide. L’application *TripReport* devra lui fournir un argument important : la durée totale des retards imputables à cet aéroport.

L'application *TripReport* possède une collection de "Mouvement" nommée "lesMouvements". Il faut donc maintenant lui ajouter une méthode "cumulRetard" prenant en paramètre le code OACI d’un aéroport et retournant le cumul des retards dont l'aéroport est responsable, sa signature est la suivante :

int cumulRetard(String oaci)

## Votre mission

**2.1 Rédiger la méthode "getImpliqueAeroport" de la classe "Retard".**

**2.2 Rédiger la méthode "dureeReelle" de la classe "Mouvement".**

**2.3 Rédiger la méthode "dureeRetardResponsable" de la classe "Mouvement".**

**2.4 Rédiger la méthode "cumulRetard" de l'application "TripReport".**

**Mission 3 – Gérer les équipages**

Documents à utiliser : 7 et 8

*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).*

Lors d’un vol (mouvement), le pilote doit vérifier avant le décollage que son équipage est au complet. Actuellement une liste papier, remise au pilote avant le départ, lui permet de connaître la composition de son équipage. La compagnie *Aéroplan* souhaite intégrer cette vérification à l’application *TripReport*.

Les membres d’équipage appartiennent à deux catégories distinctes :

* Les personnels navigants techniques qui s’occupent du pilotage de l’avion.
* Les personnels navigants commerciaux qui sont en cabine.

Lors d'un vol, chaque membre d’équipage occupe un des rôles suivants :

Pour les personnels navigants techniques :

* Commandant de Bord (CDB)
* Officier Pilote de Ligne (OPL)

Pour les personnels navigants commerciaux :

* Chef de Cabine Principal (CCP)
* Chef de Cabine (CC)
* Hôtesse ou steward (H/S)

La compagnie *Aéroplan* fixe des règles très strictes en matière de sécurité et de confort des passagers, au-delà des exigences légales de l’aviation civile. Elle définit donc pour chaque type d’avion un nombre minimum de pilotes, d’hôtesses et stewards.

## Votre mission

**3.1 Proposer une modélisation de la nouvelle base de données à mettre en place en intégrant la base existante et les nouveaux besoins en information. L’ensemble de la solution proposée devra être cohérent avec le formalisme choisi.**

**Mission 4 – Étude prévisionnelle de l'indemnisation des retards**

Documents à utiliser : 7, 9 et 10

**D**ans une démarche globale d’amélioration de la qualité, la compagnie *Aéroplan* accorde une grande importance à l’analyse des retards. Grâce aux informations issues de l’application *TripReport* centralisées dans la base de données, on envisage d’établir la liste ci-dessous.

**Liste des retards du jour**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N° de vol** | **Raison du retard** | **Durée** | **Nombre de passagers** | **Distance** |
| RLG-6263 RLG-6263 XMA-H712  XMA-H712 ZAP-F040 | Piste occupée Vent défavorable Passagers en retard Trafic  Trafic | 30 10  5 10 10 | 132 132 98 98 107 | 627 627 520 520 710 |

Un vol est listé si son départ a eu lieu la veille de l’édition de la liste.

**P**our analyser ces retards, l’interrogation de la base de données permet d’obtenir le tableau de bord "Ventilation des retards" présenté ci-dessous :

**Distance en kilomètres**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Durée vol** | **nombre de vols en retard**  **mars 2014** | [0 ; 1500[ | [1500 ; 3500[ | [3500 et plus |
| [1 ; 15[ | 10 | 20 | 50 |
| [15 ; 120[ | 2 | 0 | 0 |
| [120 ; 180[ | 0 | 5 | 3 |
| [180 ; 240[ | 4 | 0 | 5 |
| [240 et plus | 1 | 0 | 0 |

**L**e règlement européen 261/2004 établit des règles communes en matière d'indemnisation et d'assistance des passagers en cas de refus d'embarquement, d'annulation ou de retard important d'un vol.

Les règles d'indemnisation prévues par le règlement 261/2004 établissent qu’un retard est considéré comme important au-delà de 3 heures et entraîne le versement d'une indemnité individuelle dont le montant est fixé à :

a) 250 € pour tous les vols de 1 500 kilomètres ou moins,

b) 400 € pour tous les vols intracommunautaires de plus de 1 500 kilomètres et pour tous les autres vols de 1 500 à 3 500 kilomètres,

c) 600 € pour tous les vols qui ne relèvent pas des points a) et b).

La compagnie *Aéroplan* doit être en mesure de connaître l'indemnisation prévisionnelle globale en cas de retard compte tenu des renseignements transmis par les pilotes via l'application *TripReport* installée sur les tablettes. Pour répondre à ce besoin, le choix a été fait de créer une fonction stockée sur la base de données centralisée afin d'effectuer un traitement côté serveur.

Vous disposez pour cela de la procédure stockée permettant de valoriser ses cinq paramètres de sortie :

**get\_param\_indemnites (seuil\_retard OUT INTEGER, montant1 OUT REAL, montant2 OUT REAL, montant3 OUT REAL, distance1 OUT INTEGER, distance2 OUT INTEGER)**

* *seuil\_retard : nombre de minutes à partir duquel une indemnité est due (actuellement 180 min. soit 3 heures)*
* *montant1 : montant de l’indemnité de retard pour les vols de courte distance (actuellement 250 €)*
* *montant2 : montant de l’indemnité de retard intermédiaire (actuellement 400 €)*
* *montant3 : montant de l’indemnité la plus forte (actuellement 600 €)*
* *distance1 : actuellement 1 500 kilomètres*
* *distance2 : actuellement 3 500 kilomètres*

## Votre mission

**4.1 Écrire la requête permettant d’extraire les informations nécessaires à l’obtention de la liste des retards du jour.**

**4.2 Expliquer la signification de la valeur 2 contenue dans la case grisée du tableau de bord "Ventilation des retards".**

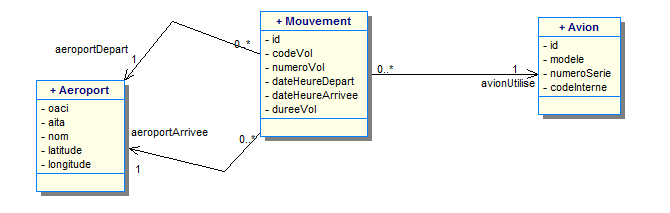
**4.3 En utilisant la vue "volsRetardes", écrire la requête permettant d’obtenir la valeur contenue dans la case grisée du tableau de bord "Ventilation des retards", pour le mois de mars 2014.**

**4.4 Écrire la fonction stockée "get\_indemnisation\_globale*"* qui retourne le montant global des indemnités d'un mouvement dont l'ID est passé en paramètre.**

**4.5 À partir de la structure de la base de données, proposer deux nouveaux indicateurs permettant d’affiner l’analyse des retards, en expliquant leur mode de calcul.**

**Dossier documentaire**

Document 1 – Extrait du diagramme de classes de l'application



Remarque : Les méthodes ne sont pas représentées sur le diagramme.

Document 2 – Description des classes métier (extrait)

|  |
| --- |
| **public** **class** Aeroport {  **private** String oaci; *//le code oaci identifie l’aéroport en tant que terrain d'aviation*  **private** String aita; *//le code aita identifie l'aéroport en tant qu'aéroport commercial*  **private** String nom;  **private** String latitude;  **private** String longitude;    *// constructeur qui valorise les attributs privés*  **public** Aeroport(String oaci, String aita, String nom, String latitude,  String longitude) {...}    *// retourne le code oaci*  **public** String getOaci() {  return oaci ;  }  } |

|  |
| --- |
| **public** **class** TypeRetard {  **private** int id;  **private** String codeSituation; *// AV :avant vol, EV : en vol, AP : après vol*  **private** String libelle;    *// constructeur qui valorise les attributs privés*  **public** TypeRetard(int id, String codeSituation, String libelle) {...}  *// retourne le code de la situation*  **public** String getCodeSituation() {  return codeSituation;  }  } |

|  |
| --- |
| public class Retard {  **private** int id ;  **private** String commentaire;  **private** TypeRetard type;  **private** int duree; *//Durée du retard exprimée en minutes*    *// constructeur qui valorise les attributs privés*  **public** Retard(int id, String commentaire, TypeRetard type, int duree) {...}    *// retourne le type de retard*  **public** TypeRetard getType() {  return type ;  }  *// retourne la durée du retard*  **public** int getDuree() {  return duree ;  }  } |

|  |
| --- |
| **public** **class** Mouvement {  **private** int id;  **private** String codeVol;  **private** String numeroVol;  **private** DateHeureFr dateHeureDepart; *//date et heure de depart prévues*  **private** DateHeureFr dateHeureArrivee; *//date et heure d'arrivée prévues*  **private** int dureeVol ; *// la durée du vol est exprimée en minutes*  **private** Avion avionUtilise;  **private** Aeroport aeroportDepart;  **private** Aeroport aeroportArrivee;  **private** ArrayList<Retard> lesRetards;  *// constructeur qui valorise les attributs privés et instancie la collection lesRetards*  **public** Mouvement(int id, String codeVol, String numeroVol,  DateHeureFr dateHeureDepart, DateHeureFr dateHeureArrivee,  int dureeVol, Avion avionUtilise, Aeroport aeroportDepart,  Aeroport aeroportArrivee) {...}  *// retourne l’aéroport de départ*  **public** Aeroport getAeroportDepart() { return aeroportDepart; }  *// retourne l’aéroport d’arrivée*  **public** Aeroport getAeroportArrivee() { return aeroportArrivee; }  *// Ajoute un retard au Mouvement*  **public** void ajouteRetard(Retard r) { lesRetards.add(r); }  *// Calcule le retard total du mouvement*  **public** int retardTotal() {  int res = 0;  for(Retard r : lesRetards) {  res = r.getDuree();  }  return res;  }  } |

Document 3 – Exemple d'utilisation d'une collection

L’exemple ci-dessous permet de manipuler une collection de chaînes de caractères. Le principe est le même quel que soit le type des éléments.

ArrayList<String> mesChaines; *// déclaration d'une collection de chaînes de caractères*

mesChaines = new ArrayList<String>(); *// instanciation de la collection*

mesChaines.add("un"); *// ajout d'une chaîne à la collection*

mesChaines.add("deux");

mesChaines.add("trois");

for (String uneChaine : mesChaines) { *// parcours de la collection*

System.out.println(uneChaine); *// affichage de l'élément courant*

}

mesChaines.remove(1); *// suppression du 2ème élément (indice 1)*

System.out.println(mesChaines.get(0)); *// affichage du 1er élément (indice 0)*

System.out.println(mesChaines.size()); *// affichage du nombre d'éléments de la collection*

Document 4 – Ajout de tests unitaires pour la classe "Mouvement"

|  |
| --- |
| **public** **class** MouvementTest {  **private** Mouvement mvt;  //La methode "setup()" est automatiquement exécutée avant chaque appel de méthode de test  @Before  **public** **void** setUp() **throws** Exception {  Avion avion = **new** Avion("FGFKA", "A319");  Aeroport aeroD = **new** Aeroport("LFBD", "BOD", "Bordeaux-Mérignac", “44-50N”, “000-42W”);  Aeroport aeroA = **new** Aeroport("LFLL", "LYS", "Lyon-Saint-Exupéry", “45-44N“,“005-05E“);  DateHeureFr dateD = **new** DateHeureFr(17, 7, 2013, 18, 30);  DateHeureFr dateA = **new** DateHeureFr(17, 7, 2013, 19, 45);    mvt = **new** Mouvement(1221, "RLG", "7410", dateD, dateA, 75, avion, aeroD, aeroA);  }  @Test  **public** **void** testAjouteRetardInitial() {  TypeRetard leType = **new** TypeRetard(1, "AV", "Occupation des pistes");  Retard unRetard = **new** Retard(1,"pluie au décollage", leType, 50);    mvt.ajouteRetard(unRetard);  *assertEquals*("1er retard non ajouté", 1, mvt.getLesRetards().size());    Retard premRetard = mvt.getLesRetards().get(0);  *assertEquals*("1er retard non accessible", unRetard, premRetard);  }    @Test  **public** **void** testAjouteRetardSuivant() {  TypeRetard leType1 = **new** TypeRetard(1, "AV", "Occupation des pistes");  TypeRetard leType2 = **new** TypeRetard(2, "EV", "Situation climatique");  Retard unRetard = **new** Retard(1,"pluie au décollage", leType1, 50);  Retard autreRetard = **new** Retard(2,"Vent défavorable", leType2, 20);    mvt.ajouteRetard(unRetard);  mvt.ajouteRetard(autreRetard);  *assertEquals*("Retard suivant non ajouté", 2, mvt.getLesRetards().size());    Retard retardSuiv = mvt.getLesRetards().get(1);  *assertEquals*("Retard suivant non accessible", autreRetard, retardSuiv);  }  } |

Document 5 – Fiche d'incident n°D24

|  |  |  |  |
| --- | --- | --- | --- |
| Description de l’incident | | | |
| Date signalement :  01/08/2013 | Rédacteur :  Arnaud Belloir (Pilote) | Niveau de gravité :  Bloquant Majeur Mineur | |
| Environnement concerné | Matériel : Tablette | Logiciel :TripReport | |
| Description du problème avec éventuelles captures d'écran, messages d'erreurs : | | | |
| Lorsque j'affiche le récapitulatif du mouvement créé, le retard total est faux. J’obtiens les informations suivantes : | | | |
| Cause du problème | | | |
| Date analyse : | Rédacteur : | | |
| Description de la cause : | | | |
|  | | | |
| Actions réalisées | | | |
| Date réalisation : | Rédacteur : | | Version de correction : |
|  | | | |

Document 6 – Évolution des classes "Mouvement" et "Retard"

**Ajouts à la classe Retard :**

**public class** Retard {

*// Précise si l’aéroport est responsable du retard*

**private boolean** impliqueAeroport ;

…

**public boolean** getImpliqueAeroport() { … }

}

**Ajouts à la classe Mouvement :**

**public class** Mouvement {

*// Retourne la durée réelle du mouvement : durée prévue du vol + somme des retards*

**public int** dureeReelle() { …. }

*// Reçoit en paramètre une situation représentée par le code ‘AV’ (Avant Vol) si on veut*

*// Calculer le retard dont l’aéroport de départ est responsable, ou le code ‘AP’ (Après*

*// Vol) si on veut calculer le retard dont l’aéroport d’arrivée est responsable*

*// Retourne la durée totale des retards du mouvement dans cette situation pour*

*// lesquels l’aéroport est responsable*

**public** **int** dureeRetardResponsable(String situation) { …. }

}

Document 7 – Présentation de la base de données

**Modélisation conceptuelle**

***La modélisation conceptuelle est fournie dans les deux représentations les plus courantes.***

|  |  |
| --- | --- |
| **Schéma entité-association** | **Diagramme de classes** |
|  |  |

**Schéma relationnel**

**TYPERETARD** (id, nom)

Clé primaire : id

**AEROPORT** (OACI, AITA, nom, latitude, longitude)

Clé primaire : OACI

**AVION** (id, modele, numeroSerie, codeInterne)

Clé primaire : id

**MOUVEMENT** (id, idAvion, numeroVol, idAeroportDepart, distance, nbPassagers, estIntracom, dateHeureDepart, idAeroportArrivee, dateHeureArrivee, dureeVol)

Clé primaire : id

Clés étrangères : idAvion en référence à id de AVION

idAeroportDepart en référence à OACI de AEROPORT

idAeroportArrivee en référence à OACI de AEROPORT

**RETARD** (id, raison, duree, idType, idMvt)

Clé primaire : id

Clés étrangères : idType en référence à id de TYPERETARD

idMvt en référence à id de MOUVEMENT

* Tous les aéroports du monde sont désignés par un code unique distribué par l’organisation de l'aviation civile internationale (OACI). Les aéroports commerciaux ont en plus un code donné par l’association internationale du transport aérien (AITA).
* L'attribut **id** de la table MOUVEMENT est de type INTEGER.
* L'attribut **distance** de la table MOUVEMENT est exprimé en kilomètres.
* Les attributs **dateHeureDepart** et **dateHeureArrivee** représentent les dates et heures prévues.
* L'attribut **estIntracom** de la table MOUVEMENT est valorisé à 1 s'il s'agit d'un vol intracommunautaire et à 0 sinon. Un vol intracommunautaire est un vol entre deux villes de l'union européenne.
* L'attribut **duree** de la table RETARD est de type INTEGER et exprimé en minutes.

**Vue "volsRetardes"**

La vue "**volRetardes**" a été créée pour réaliser diverses statistiques :

**CREATE VIEW volsRetardes**

AS

SELECT idMvt, distance, nbPassagers, estIntracom , dateHeureDepart,

SUM(duree) As retardTotal

FROM RETARD INNER JOIN MOUVEMENT

ON RETARD.idMvt=MOUVEMENT.id

GROUP BY idMvt, distance, nbPassagers, estIntracom, dateHeureDepart

Exemple du résultat de l’exécution de la requête "select \* from volsRetardes" :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| idMvt | distance | nbPassagers | estIntracom | dateHeureDepart | retardTotal |
| 456 | 710 | 107 | 0 | 2014/02/18-12:23 | 220 |
| 672 | 500 | 95 | 1 | 2014/02/19-08:06 | 120 |
| … | … | … | … | … | … |

Document 8 – Documents concernant les équipages

Extrait de la liste des modèles d’avions d'Aéroplan avec, pour chacun, le nombre minimum réglementaire de personnel par rôle.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Modèle d’avion** | **Nombre de sièges** | **CDB** | **OPL** | **CCP** | **CC** | **H/S** |
| A319 | 142 | 1 | 1 |  | 1 | 2 |
| A332 | 208 | 1 | 1 |  | 1 | 3 |
| A343 | 275 | 1 | 2 | 1 | 1 | 4 |
| A380 | 516 | 2 | 2 | 1 | 2 | 18 |
| B772 | 247 | 1 | 1 |  | 1 | 4 |

Exemples de listings de personnel remis aux pilotes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mouvement : Vol RLG-6263 du 28/07/2013 - Avion : F-GMPD de type A319 | | | | |
| CDB : | - FABRE François | 06 50 19 78 62 | 10 590 | heures de vol |
| OPL : | - LEMAITRE Hervé | 06 40 96 35 97 | 2 500 | heures de vol |
| CC : | - LEFEVRE Martine | 06 84 79 65 84 | Anglais, Français, Espagnol | |
| H/S : | - MARTY Paul | 06 74 93 65 84 | Anglais, Français | |
|  | - FONTAINE Emma | 06 94 47 63 26 | Anglais, Français | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mouvement : RLG-4692 du 28/07/2013 - Avion : F-GLZI de type A343 | | | | |
| CDB : | - MORIN Marie | 06 30 26 97 36 | 53 700 | heures de vol |
| OPL : | - PERROT Alexandre | 06 25 93 84 75 | 20 630 | heures de vol |
|  | - FABRE François | 06 50 19 78 62 | 10 590 | heures de vol |
| CCP | - DUVAL Franck | 07 40 93 24 51 | Anglais, Français, Espagnol, Allemand | |
| CC : | - GAUTIER Emeline | 06 84 79 65 84 | Anglais, Français, Espagnol, Allemand | |
| H/S : | - LEFEVRE Martine  - REYNAUD Lina | 06 84 79 65 84  06 94 47 63 26 | Anglais, Français, Espagnol  Anglais, Français | |
|  | - MARTY Paul | 06 74 93 65 84 | Anglais, Français | |
|  | - MASSON Jean-Pierre | 06 32 95 67 46 | Anglais, Français, Portugais | |

Remarques :

* Pour chaque membre d’équipage, il faut enregistrer le numéro de portable professionnel afin de pouvoir le joindre rapidement en cas de besoin.
* Il est nécessaire de connaître le nombre d’heures de vol des personnels navigants techniques.
* Il est nécessaire de connaître toutes les langues parlées par les personnels navigants commerciaux.

Document 9 – Fonctionnalités du SGBD

**Les fonctions pour manipuler les dates :**

Le SGBD fournit les fonctions suivantes :

DATE() qui permet d'obtenir la date système.

YESTERDAY() qui permet d'obtenir la date précédent la date système.

YEAR(uneDate) qui permet d'extraire l'année d'un champ DATE.

YEAR(uneDateHeure) qui permet d'extraire l'année d'un champ DATEHEURE.

MONTH(uneDate) qui permet d'extraire le mois d'un champ DATE.

MONTH(uneDateHeure) qui permet d'extraire le mois d'un champ DATEHEURE.

GETDATE(uneDateHeure) qui permet d'extraire la date d'un champ DATEHEURE.

**Les fonctions stockées :**

La fonction stockée suivante permet de renvoyer la distance d'un mouvement en kilomètres ou en miles nautiques connaissant l'ID d'un mouvement et l'unité choisie ('k' ou 'm')

*/\* Cette ligne indique que la fonction reçoit deux paramètres en entrée \*/*

CREATE FUNCTION la\_distance (id\_mouvement IN INTEGER, unite IN CHAR)

*/\* La section DECLARE permet de déclarer les variables \*/*

DECLARE

resultat REAL;

dist REAL;

BEGIN

*/\*récupération de la distance et affectation de la variable dist \*/*

SELECT distance INTO dist

FROM MOUVEMENT

WHERE id=id\_mouvement;

resultat := dist;

IF (unite='m') THEN

*/\* 1 mile nautique correspond à 1.852 kilomètre \*/*

resultat := dist / 1.852;

END IF

RETURN resultat;

END

Document 10 – Fonction stockée "get\_indemnisation\_globale"

CREATE FUNCTION get\_indemnisation\_globale (id\_mouvement IN INTEGER)

DECLARE

*/\* variables valorisées par l’appel de get\_param\_indemnites \*/*

p\_montant1, p\_montant2, p\_montant3 REAL;

p\_seuil\_retard, p\_distance1, p\_distance2 INTEGER;

*/\* déclaration des autres variables locales \*/*

var\_distance REAL;

var\_nb\_passagers INTEGER;

var\_intra\_com SMALLINT;

var\_retard INTEGER;

var\_indemnite REAL;

BEGIN

get\_param\_indemnites(p\_seuil\_retard, p\_ montant1, p\_montant2,

p\_montant3, p\_distance1, p\_distance2);

**/\* code à compléter \*/**

RETURN var\_indemnite

END