BTS SERVICES INFORMATIQUES AUX ORGANISATIONS

E5 : Production et fourniture de services informatiques

SESSION 2016

Durée : 4h00 Coefficient : 5

ÉTUDE DE CAS

CAS DÉMGUIV

Ce sujet comporte 17 pages dont un dossier documentaire de 11 pages. La candidate ou le candidat est invité.e à vérifier que le sujet qui lui a été remis est complet.

Conformément aux recommandations du Haut Conseil à l'Égalité entre les femmes et les hommes dans son guide publié en novembre 2015, l'expression du féminin et du masculin s'effectue en utilisant le point, par exemple, client.e.

Aucun matériel ni document autorisé

Dossier documentaire :

Barème

Mission 1	Améliorer la planification des déménagements	20 points
Mission 2	Gérer les véhicules de déménagement	30 points
Mission 3	Gérer les absences des salariés	30 points
Mission 4	Suivre la réalisation des déménagements	20 points
	Total	100 points

PRÉSENTATION DU CONTEXTE

***L'organisation cliente :* DÉMGUIV**

DÉMGUIV est une entreprise familiale spécialisée dans le déménagement de particuliers interurbain et intra-urbain. Fondée par M. GUIVARCH en 1980 dans la ville de Quimper, la société DÉMGUIV n'a cessé de se développer et compte aujourd'hui trois nouvelles agences : Rennes, Nantes et Angers. Le siège social de la société est situé à Quimper dans les mêmes locaux que l'agence. Chaque agence est dirigée par un responsable d'agence.

La société DÉMGUIV propose plusieurs formules de déménagement réservées aux particuliers :

- La formule économique dans laquelle les déménageurs s'occupent uniquement du chargement et de la livraison des meubles et des cartons.
- La formule standard comprend toutes les prestations de la formule économique avec démontage et remontage des meubles si cela s'avère nécessaire. Les cartons sont fournis par l'entreprise.
- La formule de luxe comprend toutes les prestations de la formule standard en plus de la mise en cartons et du déballage.
- La formule sur mesure comprend toutes les prestations de l'une des trois formules précédentes au choix de la cliente ou du client, enrichies par ses besoins spécifiques. Par exemple, la cliente ou le client peut choisir la formule économique en déléguant l'emballage des objets fragiles à l'entreprise DÉMGUIV.

***L'entreprise prestataire de services :* LOGIMOUVE**

LOGIMOUVE est une entreprise de services du numérique (ESN) nantaise spécialisée dans le développement de logiciels destinés aux professionnels du transport. Elle assure, en outre, la formation et l'assistance pour sa gamme de logiciels. Fondée en 2003, elle compte quinze salariés dont dix développeurs.

Le projet

Afin d'optimiser la gestion de ses contrats de déménagements, la société DÉMGUIV a fait appel en 2013 à un prestataire informatique pour se doter d'un logiciel de gestion des déménagements. En 2014, le logiciel *PSDEM (Planification-Suivi Déménagement)* a été déployé dans les agences de DÉMGUIV.

L'application *PSDEM* est développée en langage *Java*. Elle est composée :

- d'un module de gestion des dossiers de déménagements : *SUIVI_DEM* ;
- d'un module de planification des déménagements : *PLAN_DEM* ;
- d'un module de gestion des absences des salariés : *GEST_ABS*.

L'application *PSDEM* est installée dans les quatre agences et s'appuie sur une base de données située à Quimper accessible depuis toutes les agences.

L'utilisation de *PSDEM* a révélé des dysfonctionnements qui n'ont pu être corrigés suite à la liquidation judiciaire de son éditeur initial. L'entreprise DÉMGUIV a alors confié la correction et l'évolution de l'application à l'ESN LOGIMOUVE.

Nouvellement embauché.e dans l'entreprise LOGIMOUVE, vous êtes chargé.e de participer aux différentes missions du projet d'évolution de l'application *PSDEM*.

Mission 1 : Améliorer la planification des déménagements

Documents à utiliser : 1, 2, 3 et 4

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).

Lorsqu'un.e client.e potentiel.le contacte une agence de DÉMGUIV pour une demande de déménagement, celle-ci est transmise au responsable d'agence qui confie l'affaire à un agent commercial.

L'agent commercial prend contact avec la cliente ou le client pour convenir d'une date de visite de son logement actuel. Cette visite permet à l'agent commercial :

- d'évaluer précisément le volume du mobilier à déménager ainsi que les conditions d'accès au chargement comme au déchargement (passage poids lourd, rue piétonne, conditions de stationnement …) ;
- de présenter les différentes formules que peut proposer son entreprise.

À l'issue de cette visite, l'agent commercial envoie un ou plusieurs devis à la cliente ou au client, selon les prestations et le calendrier fixés avec ce dernier. Quand un devis est accepté, le dossier est confié au chef d'exploitation de l'agence, responsable de la planification du déménagement.

Le chef d'exploitation analyse les informations liées au dossier du déménagement puis :

- décide du ou des véhicule(s) qui seront utilisés pour le déménagement ;
- affecte le nombre de déménageurs nécessaires ;
- nomme un chef d'équipe qui sera le seul point de liaison entre la cliente ou le client et l'entreprise pendant toute la durée de l'opération.

Pour organiser les différents déménagements, les chefs d'exploitation (un par agence) s'appuient sur le module *PLAN_DEM de* l'application logicielle *PSDEM*. Ils ont constaté quelques dysfonctionnements.

Le chef d'exploitation de l'agence de Quimper a été désigné pour rédiger un document, à destination de LOGIMOUVE qui synthétise les différents dysfonctionnements et les évolutions souhaitées du module.

Afin de répondre aux nouveaux besoins exprimés par les chefs d'exploitation des agences de DÉMGUIV, votre collègue chargé du développement des nouvelles fonctionnalités vous demande de participer au travail concernant les anomalies constatées et les évolutions souhaitées.

Votre mission

1. **Réaliser les modifications nécessaires à la correction de l'anomalie *Ano_chef1*.**
2. **Rédiger une courte note indiquant le principe d'une solution permettant de corriger l'anomalie *Ano_chef2*.**
3. **Réaliser les modifications à apporter à la base de données pour satisfaire la demande d'évolution *Évo_arrêts*.**
4. **Réaliser les modifications à apporter à la base de données pour satisfaire la demande d'évolution *Évo_conduire*.**

Mission 2 : Gérer les véhicules de déménagement

Documents à utiliser : 1, 5, 6, et 7

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).

La gestion des véhicules de déménagement est à ce jour très incomplète ; il a été décidé de l'étendre.

DÉMGUIV utilise plusieurs types de véhicules. Les caractéristiques d'un type de véhicule font qu'il est nécessaire de posséder un permis de conduire spécifique pour l'utiliser. Par exemple, il faut un permis C pour conduire un véhicule de type Renault Midlum 180.12.

Il y a quatre permis de conduire : B, C1, C et CE et les possibilités se cumulent : on peut conduire un véhicule nécessitant le permis B avec le permis C1 (mais pas l'inverse), on peut conduire un véhicule nécessitant le permis C1 avec le permis CE… Les permis sont ainsi ordonnés : B, C1, C, CE. Il serait intéressant de mémoriser l'ensemble des permis possédés par un déménageur avec, pour chacun d'eux, la date à laquelle il a été obtenu.

Chaque agence DÉMGUIV abrite un atelier qui assure l'entretien des véhicules qui y sont affectés. En ce qui concerne les réparations, même si la plupart sont assurées dans les ateliers de DÉMGUIV, certaines nécessitent le recours à un prestataire externe (garage spécialisé en poids lourds).

La législation impose que chaque véhicule poids lourd soit soumis à un contrôle technique annuel. À l'issue de ce contrôle, le résultat peut être :

A : véhicule accepté
X : renvoi du véhicule sans réalisation de la visite
C : commentaire
O : observation (défaut à corriger mais sans obligation de contre-visite)
S : contre-visite avec autorisation de circuler
R : contre-visite avec interdiction de circuler

On trouve finalement trois motifs d'immobilisation d'un véhicule : entretien, réparation et contrôle technique. Ces immobilisations sont actuellement reportées sur la fiche de suivi de chaque véhicule. Il serait nécessaire de les gérer dans le module *PLAN_DEM*.

Le nouveau service à développer devra satisfaire l'ensemble des besoins concernant la gestion des véhicules et de leurs immobilisations. Vous êtes chargé.e d'étudier les modifications à apporter à la base de données pour la réalisation de cette gestion.

Une liste de besoins prévisibles en information a été établie de manière à vérifier que la future base de données pourra les satisfaire. Votre collègue initialement prévu pour traiter cette liste de besoins a été appelé sur une autre mission. Il a néanmoins eu le temps de commencer et vous a laissé une note concernant ce travail ainsi qu'un exemple représentatif de fiche de suivi véhicule.

Votre mission

5. **Indiquer pour chacun des besoins exprimés restant à traiter (REQ_003 à REQ_005), si la base de données actuelle permet de le satisfaire. Si oui, présenter la requête SQL nécessaire en se référant à la note concernant les besoins en information. Si non, indiquer brièvement la raison de cette impossibilité.**
6. **Proposer une modélisation de la nouvelle base de données à mettre en place en intégrant la base existante, la gestion des permis, des véhicules et la fiche de suivi. Seuls les éléments du schéma existant qui sont concernés par l'évolution seront repris dans le schéma proposé.**

Mission 3 : Gérer les absences des salariés

Documents à utiliser : 8, 9, 10 et 11

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.

Le module *GEST_ABS* alimente la base de données en ce qui concerne les absences des salariés. Sa version actuelle est très sommaire, une nouvelle version est à l'étude.

La durée d'une absence est exprimée en nombre de jours ouvrables. Certaines absences (exemple : congés annuels) nécessitent une demande d'autorisation que la ou le salarié.e doit formuler. D'autres absences, comme les arrêts de travail, ne nécessitent pas de demande préalable.

La nouvelle version du module *GEST_ABS* permettra :

- aux salariés, de saisir leurs demandes d'autorisation d'absence, de visualiser l'état de leurs demandes (en attente, acceptée ou refusée), de visualiser l'historique de leurs absences et de consulter l'état de leurs différents soldes ;
- aux secrétaires, de saisir les arrêts de travail des salariés ;
- aux responsables d'agence, de consulter les demandes d'autorisation d'absences et d'y répondre, de consulter l'état des soldes, de visualiser les absences sous la forme de plannings hebdomadaires et mensuels ;
- aux chefs d'exploitation, de visualiser les absences des déménageurs sous la forme de plannings hebdomadaires et mensuels.

Un planning d'absences fait apparaître de manière claire (un code couleur a été défini pour représenter les motifs d'absence) :

- les absences qui ne nécessitent pas de demande d'autorisation ;
- les absences dont la demande d'autorisation est en attente ;
- les absences dont la demande d'autorisation est acceptée.

L'itération courante du projet a pour objectif la visualisation du planning mensuel des absences.

Votre mission

7. **Fournir le code de la méthode *ajouterAbsence* de la classe *Salarie*.**
8. **Fournir le code de la méthode *refuser* de la classe *DemandeAbsence*.**
9. **Fournir le code de la méthode *getAbsencesEnAttente* de la classe *Salarie*.**
10. **Fournir le code de la méthode *getJoursAbsence* de la classe *Absence*.**

Mission 4 : Suivre la réalisation des déménagements

Documents à utiliser : 12, 13, 14, 15 et 16

La réalisation d'un déménagement est accompagnée de documents obligatoires confiés au chef d'équipe du déménagement (représentant de la société auprès de la cliente ou du client) :

- la lettre de voiture qui reprend les informations du dossier (une copie pour chaque véhicule utilisé) ;
- la fiche d'instruction et d'exécution (une copie pour chaque véhicule) ;
- la déclaration de fin de travail sur laquelle la cliente ou le client et le chef d'équipe peuvent formuler des observations.

L'entreprise DÉMGUIV souhaite automatiser cette procédure et dématérialiser les documents associés. Le chef d'équipe du déménagement sera alors équipé d'un terminal mobile fonctionnant sous *Android*.
M. Guivarch (fondateur de la société DÉMGUIV) se pose des questions sur les avantages mais aussi les risques de cette dématérialisation. Il a demandé à LOGIMOUVE de l'éclairer sur ce sujet. Un de vos collègues a récupéré sur internet quelques extraits de pages et vous demande d'en faire une synthèse.

M. Guivarch souhaite profiter de cette automatisation pour se donner les moyens d'évaluer la qualité des devis réalisés par les commerciaux de DÉMGUIV. Afin d'apprécier leur fiabilité, M. Guivarch souhaite mettre en place des indicateurs pertinents et a besoin pour cela de disposer des informations suivantes : temps d'emballage lié au déménagement et cubage (nombre de mètres cubes) chargé dans chaque camion.

Il souhaite tirer profit de la nouvelle application pour obtenir ces nouvelles informations.

L'équipe de développement a fait les choix techniques suivants :

- architecture : *Rest*
- format des données échangées : *Json*
- application serveur : *Framework PHP*
- application cliente : *Java* / *API Android*

La première itération a pour objectif la réalisation des fonctionnalités associées à la fiche d'instruction et d'exécution qui permettront au responsable du déménagement de :

- consulter la fiche d'instruction et d'exécution ;
- saisir et envoyer les informations à l'application serveur après le chargement des véhicules ;
- saisir et envoyer les informations à l'application serveur après la livraison.

Votre mission

11. **Rédiger une courte note expliquant à M. Guivarch la phrase « Protéger l'information consiste à garantir sa disponibilité, son intégrité, sa confidentialité et sa traçabilité ».**
12. **Proposer un tableau destiné à alimenter la réflexion de M. Guivarch présentant les avantages et les inconvénients de la dématérialisation des documents de déménagement.**
13. **Proposer une représentation des données échangées après le chargement (au format *Json*) entre l'application cliente et l'application serveur prenant en compte les nouvelles informations demandées par M. Guivarch.**
14. **Coder la partie de la méthode *getFicheInstructionAction* du contrôleur *RealisationController* (fichier *RealisationController.php*) correspondant à l'envoi de la réponse.**

1. Présentation de la base de données (extrait)

Schéma entité-association

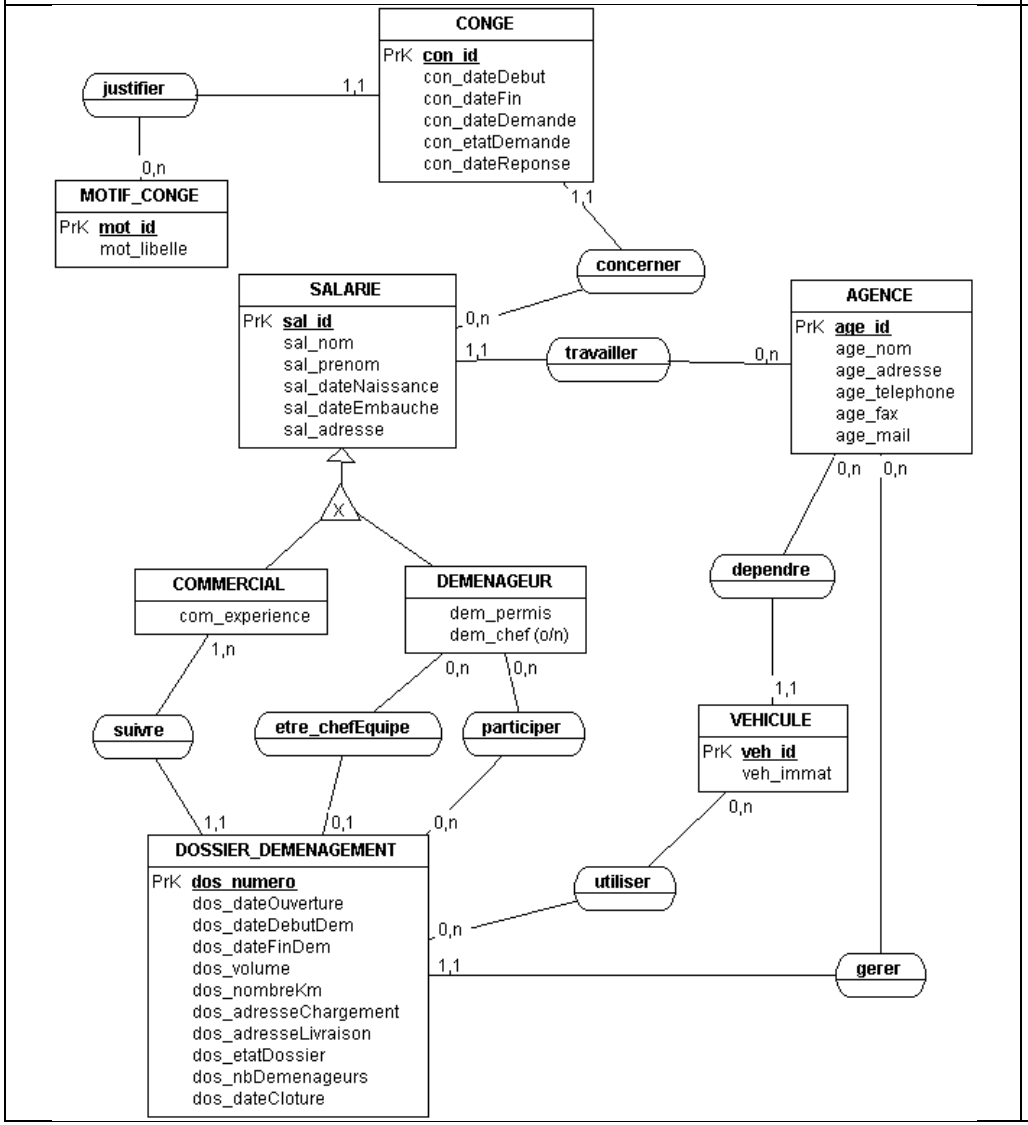


Diagramme de classes

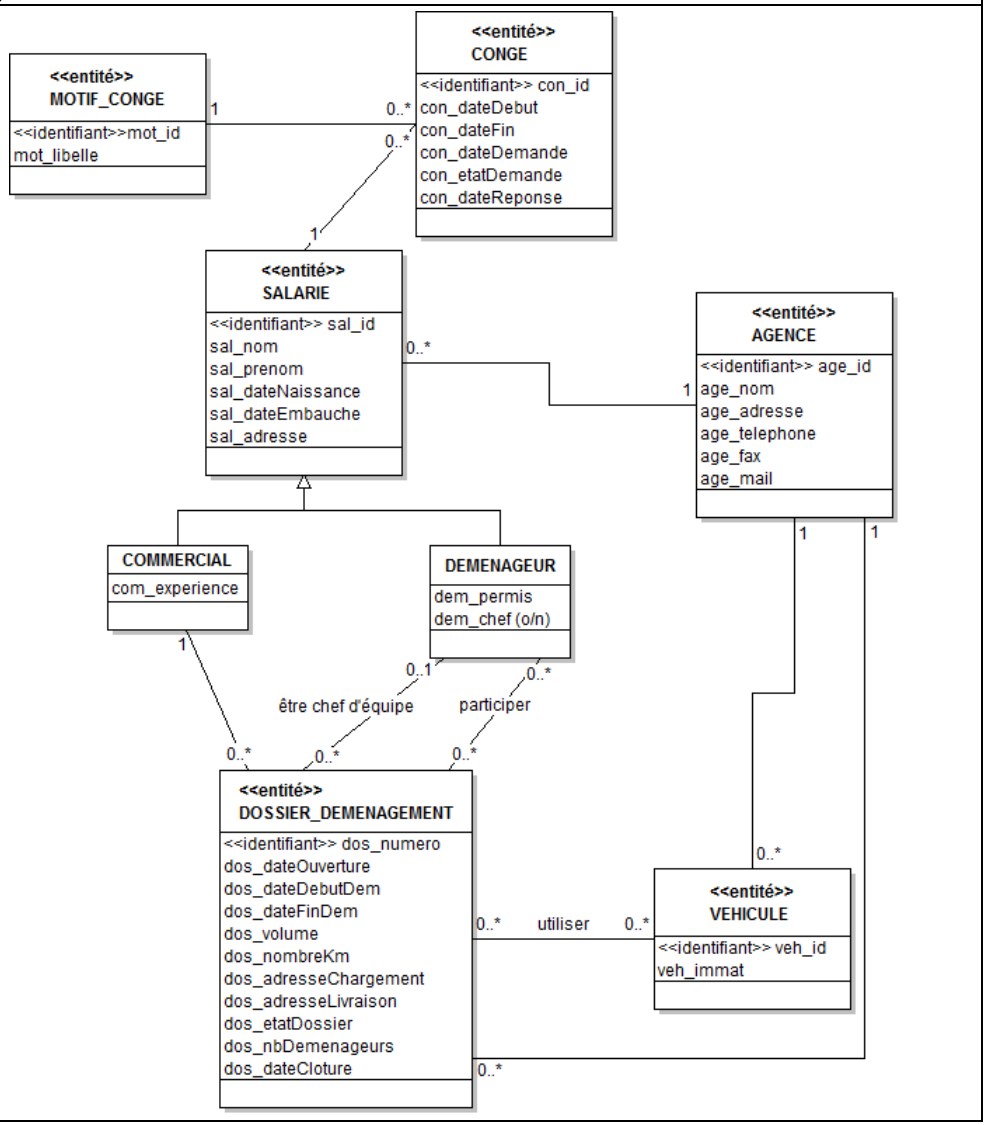


Schéma relationnel

AGENCE (age_id, age_nom, age_adresse, age_telephone, age_fax, age_mail)
Clé primaire : age_id

VEHICULE (veh_id, veh_immat, veh_agence)
Clé primaire : veh_id
Clé étrangère : veh_agence en référence à age_id de AGENCE

SALARIE (sal_id, sal_nom, sal_prenom, sal_dateNaissance, sal_dateEmbauche, sal_adresse, sal_agence, sal_type, sal_permis, sal_chef, sal_experience)
Clé primaire : sal_id
Clé étrangère : sal_agence en référence à age_id de AGENCE

/ Le champ « sal_type » contient « C » pour un commercial, « D » pour un déménageur. Le champ « sal_permis » contient le plus haut permis de conduire possédé pour un déménageur. Le champ « sal_experience » contient un indicateur de l'expérience pour un commercial. Le champ « sal_chef » est de type booléen. Il contient « 1 » si le déménageur possède la qualification nécessaire pour être désigné comme chef d'équipe responsable d'un déménagement, « 0 » sinon. */*

MOTIF_CONGE (mot_id, mot_libelle)
Clé primaire : mot_id

CONGE (con_id, con_dateDebut, con_dateFin, con_dateDemande, con_etatDemande, con_dateReponse, con_salarie, con_motif)
Clé primaire : con_id
Clé étrangère : con_salarie en référence à sal_id de SALARIE
Clé étrangère : con_motif en référence à mot_id de MOTIF_CONGE

DOSSIER_DEMENAGEMENT (dos_numero, dos_dateOuverture, dos_dateDebutDem, dos_dateFinDem, dos_volume, dos_nombreKm, dos_adresseChargement, dos_adresseLivraison, dos_etatDossier, dos_nbDemenageurs, dos_dateCloture, dos_commercial, dos_chefEquipe, dos_agence)
Clé primaire : dos_numero
Clé étrangère : dos_commercial en référence à sal_id de SALARIE
Clé étrangère : dos_chefEquipe en référence à sal_id de SALARIE
Clé étrangère : dos_agence en référence à age_id de AGENCE

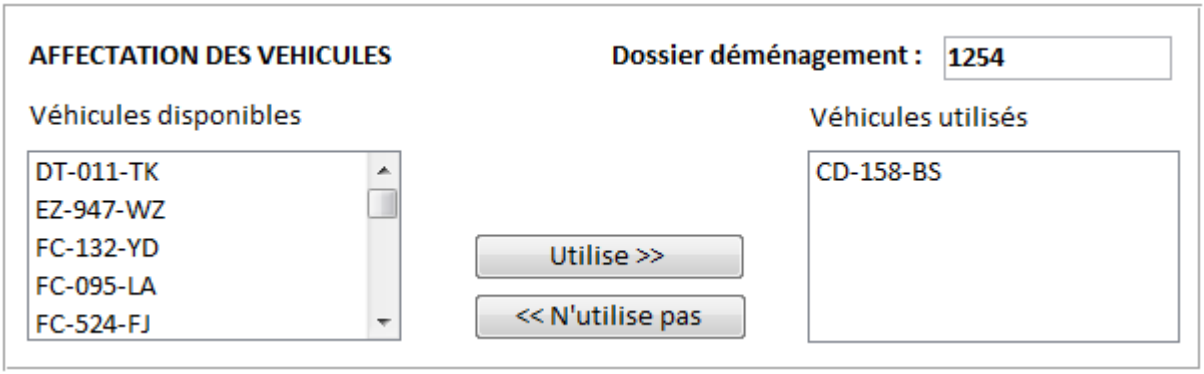
/ Le champ « dos_commercial » permet de mémoriser le commercial qui a en charge le dossier. Le champ « dos_chefEquipe » permet de mémoriser le déménageur chef d'équipe du déménagement. « dos_dateDebutDem » et « dos_dateFinDem » représentent la période prévue pour le déménagement. */*

UTILISER (uti_dossier, uti_vehicule)
Clé primaire : uti_dossier, uti_vehicule
Clé étrangère : uti_dossier en référence à dos_numero de DOSSIER_DEMENAGEMENT
Clé étrangère : uti_vehicule en référence à veh_id de VEHICULE

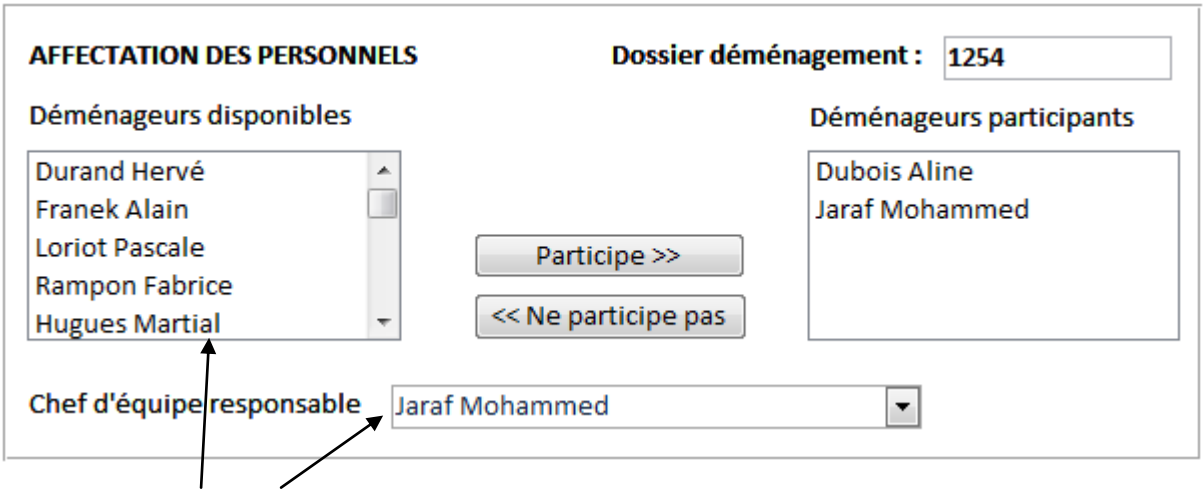
PARTICIPER (par_dossier, par_demenageur)
Clé primaire : par_dossier, par_demenageur
Clé étrangère : par_dossier en référence à dos_numero de DOSSIER_DEMENAGEMENT
Clé étrangère : par_demenageur en référence à sal_id de SALARIE

2. Éléments d'interface utilisateur du module *PLAN_DEM*

Choix des véhicules utilisés pour un déménagement :



Choix des déménageurs affectés à un déménagement :



*Ces deux listes sont alimentées par un appel à la fonction stockée « déménageursPresents » (cf document 3). Exemple : « select * from demenageursPresents(2, '25/05/2016', '30/05/2016') ».*

3. Fonction stockée de la base de données utilisée par le module *PLAN_DEM*

Cette fonction retourne l'ensemble des déménageurs de l'agence « @idAgence » qui ne sont ni en congé ni affectés à un déménagement entre les dates « @dateDebut » et « @dateFin ».

```
CREATE FUNCTION demenageursPresents(@idAgence int, @dateDebut date, @dateFin date) RETURNS TABLE
AS
RETURN      (SELECT * FROM salarie
            WHERE sal_agence = @idAgence and sal_type = 'D'
            and sal_id not in (     select con_salarie from conge
                                    where con_dateFin >= @ dateDebut and con_dateDebut <= @dateFin)
            and sal_id not in (     select par_demenageur from participer, dossier_demenagement
                                    where par_dossier = dos_numero
                                    and dos_dateFinDem >= @ dateDebut and dos_dateDebutDem <= @dateFin)
            )
```

Remarque : cette fonction a été testée et validée, elle ne comporte aucune erreur.

4. Corrections et évolutions demandées concernant le module *PLAN_DEM*

Nous utilisons l'outil informatique *PLAN_DEM* lors de la planification d'un déménagement. La première phase consiste pour nous à prévoir le ou les véhicule(s) nécessaire(s) pour un déménagement. Une fois cette phase validée, nous affectons les déménageurs, puis nous désignons le chef d'équipe responsable du déménagement. Les anomalies rencontrées ainsi que les évolutions souhaitées lors de cette planification sont résumées dans les tableaux suivants.

Anomalies détectées :

Numéro	Description
Ano_chef1	Le module *PLAN_DEM* autorise l'attribution du rôle «Chef d'équipe» à un déménageur qui ne possède pas la qualification nécessaire.
Ano_chef2	Toujours concernant le chef d'équipe, il est possible de désigner un déménageur ne participant pas au déménagement, ce qui n'a pas de sens.

Évolutions souhaitées :

Numéro	Description
Évo_arrêts	Les arrêts de travail ne sont pas gérés par *PSDEM*. Ils le sont au niveau de la gestion des ressources humaines qui nous transmet une liste des personnes en arrêt de maladie. Il faudrait intégrer cette notion dans la base de données utilisée par *PLAN_DEM*. Un arrêt de travail est différent d'un congé en ce sens qu'il n'est pas prévu. Il n'y a donc pas de demande de la part du salarié. Les seules données sont : les dates de début et de fin de l'absence, la date de prescription et la date de réception de l'arrêt de travail.
Évo_conduire	Il serait souhaitable de savoir quel(s) déménageur(s) conduiront quel(s) véhicule(s) lors d'un déménagement. Tous les déménageurs ne conduisent pas et il peut y avoir plusieurs conducteurs pour un véhicule. Un déménageur peut conduire plusieurs véhicules.

5. Besoins en information concernant la gestion des véhicules (voir également document 6)

Numéro	Situation	Besoin en information
~~REQ_001~~ (voir document 6)	Nécessité de prendre un rendez-vous pour un contrôle technique	Le véhicule immatriculé BC-095-LA est-il utilisé le 25 mai 2016 ? (on pourra afficher son nombre de déménagements à cette date, le véhicule sera disponible si ce nombre est égal à zéro)
~~REQ_002~~ (voir document 6)	Besoin urgent de contacter le chef d'équipe utilisant un véhicule	Nom et prénom du chef d'équipe du déménagement utilisant le véhicule immatriculé BC-095-LA le 12 mai 2016.
REQ_003	Planification des entretiens pour un atelier	Immatriculation des véhicules de l'agence n°1 occupés le 12 mai 2016.
REQ_004	Contrôle de la fréquence des entretiens	Combien de kilomètres le véhicule immatriculé BC-095-LA a-t-il parcouru depuis son dernier entretien ?
REQ_005	Contrôle de la fréquence des entretiens	Sachant que la donnée « dos_nombreKm » indique la distance aller/retour pour un déménagement, combien de kilomètres le véhicule immatriculé BC-095-LA a-t-il parcouru lors des déménagements commencés après le 12 février 2016 ?

6. Note concernant les besoins en information

J'ai commencé le travail en écrivant une **fonction stockée « occupationVehicules »** dont je pense qu'elle sera utile pour certaines requêtes. J'ai eu ensuite le temps de traiter les deux premières, bon courage pour la suite !

-- Cette fonction retourne toutes les informations sur l'occupation des véhicules à une date donnée.
-- Elle retourne une table temporaire et son appel peut être utilisé exactement comme une table normale.

```
CREATE FUNCTION occupationVehicules(@journee date) RETURNS TABLE
AS
RETURN (SELECT * FROM dossier_demenagement,utiliser,vehicule
    WHERE dos_numero = uti_dossier and veh_id = uti_vehicule
    and dos_dateDebut <= @journee and dos_dateFin >= @journee)
```

REQ_001 : (voir document 5)
```
select COUNT(*) from occupationVehicules('25/05/2016')
where veh_immat = 'BC-095-LA'
```

REQ_002 : (voir document 5)
```
select sal_nom, sal_prenom from salarie, occupationVehicules('12/05/2016')
where sal_id = dos_chefEquipe and veh_immat='BC-095-LA'
```

7. Exemple de fiche de suivi véhicule

FICHE DE SUIVI VÉHICULE — AGENCE : NANTES

Caractéristiques :

N° de véhicule : 21

Type : Renault Midlum 180.12

Immatriculation : BC-095-LA
Date de mise en circulation : 18/02/2012
Volume utile : 43.06 m3
Hayon : oui
Couchette : non
Nombre de places en cabine : 3

PTAC : 11T99
Fréquence entretien : 60 000 km

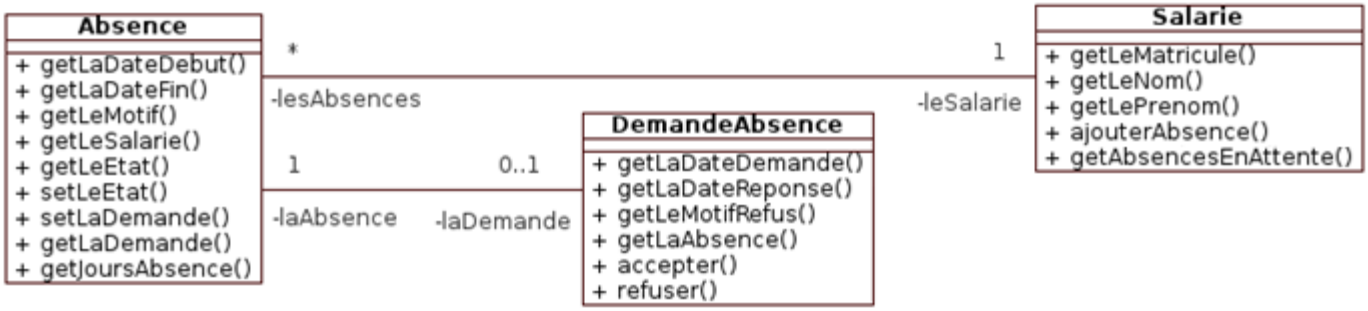
Immobilisations :

N°	Type	Kilométrage	Début	Fin	Informations supplémentaires
1	Entretien	30 128 km	02/05/2012	02/05/2012	
2	Réparation	45 302 km	18/07/2012	20/07/2012	Atelier
3	Entretien	61 382 km	30/10/2012	30/10/2012	
4	Contrôle	61 412 km	02/11/2012	02/11/2012	Véhicule accepté
5	Réparation	98 254 km	12/01/2013	15/01/2013	Renault Trucks Nantes
…	…	…	…	…	…
22	Entretien	251 403 km	02/05/2016	02/05/2016	

Remarques :

- Le volume utile, la présence d'un hayon ou d'une couchette peuvent varier entre deux véhicules de même type car ils peuvent être carrossés différemment. Le PTAC (poids total autorisé en charge), en revanche ne varie jamais.
- Dans le cas d'un contrôle technique, la dernière colonne permet d'indiquer le résultat du contrôle.
- Dans le cas d'une réparation hors atelier, la dernière colonne indique le garage ayant réparé le véhicule. Il conviendra de mémoriser le nom, l'adresse et le téléphone des garages auxquels DÉMGUIV fait appel.

8. Extrait du diagramme de classes métier



9. Description textuelle des classes métier

```
public class Salarie {
    private String leMatricule;
    private String leNom;
    private String lePrenom;
    private List<Absence> lesAbsences = new ArrayList<Absence>();

    public Salarie(String unMatricule, String unNom, String unPrenom) { … }
    public String getLeMatricule() { … }
    public String getLeNom() { … }
    public String getLePrenom() { … }
    public void ajouterAbsence(Absence uneAbsence) { … }
// Ajoute une absence à la collection lesAbsences
    public List<Absence> getAbsencesEnAttente() { … }
// Retourne la collection des absences en attente de traitement par le responsable (leEtat=ATT)
}
```

```
public class Absence {
    private DateFr laDateDebut;
    private DateFr laDateFin;
    private String leMotif;
    private String leEtat; // ATT pour en attente, ACC pour acceptée, REF pour refusée
    private Salarie leSalarie;
    private DemandeAbsence laDemande;
    public Absence(DateFr uneDateDebut, DateFr uneDateFin, MotifAbsence unMotif,
                Salarie unSalarie) { … }
    public DateFr getLaDateDebut() { … }
    public DateFr getLaDateFin() { … }
    public String getLeMotif() { … }
    public Salarie getLeSalarie() { … }
    public String getLeEtat() {    }
    public void setLeEtat(String leEtat) {    }
    public void setLaDemande(DemandeAbsence uneDemande) { … }
    public DemandeAbsence getLaDemande() { … }
    public List<DateFr> getJoursAbsence(int unMois, int uneAnnee) { … }
// Retourne la liste des jours ouvrables du mois spécifié que couvre l'absence, le principe est de
// parcourir les jours d'absence et vérifier s'ils sont dans le mois spécifié, et s'ils sont ouvrables.
// Exemple : pour une absence allant du 27/05/2016 au 06/06/2016, cette méthode appelée avec les
// paramètres unMois=6 et uneAnnee=2016 retournera la collection de dates
// { 01/06/2016, 02/06/2016, 03/06/2016, 06/06/2016} car les 4 et 5 juin 2016 ne sont pas des jours
// ouvrables.
}
```

```
public class DemandeAbsence {
    private DateFr laDateDemande;
    private DateFr laDateReponse;
    private String leMotifRefus;
    private Absence laAbsence;

    public DemandeAbsence(DateFr uneDateDebut, DateFr uneDateFin, String unMotif,
                                        Salarie unSalarie, DateFr uneDateDemande) {
        laDateDemande = uneDateDemande;
        laAbsence = new Absence(uneDateDebut, uneDateFin, unMotif, unSalarie);
        laAbsence.setLeEtat("ATT");
    }
    public DateFr getLaDateDemande() { …. }
    public DateFr getLaDateReponse() { … }
    public String getLeMotifRefus() { … }
    public Absence getLaAbsence() { … }
    public void accepter() { … }
    public void refuser(String unMotifRefus) { … }
// Constate le refus de la demande et en mémorise le motif
}
```

10. Description de la classe technique *DateFr* (extrait)

```
public class DateFr {
    public DateFr() { … } // initialisation avec la date du jour
    public DateFr(int jour, int mois, int annee) { … } // initialisation avec les valeurs des paramètres
    public int getJour() { … }
    public int getMois() { … }
    public int getAnnee() { … }
    public int getNbJoursMois() { … } // Retourne le nombre de jours que compte le mois
    public boolean estAvant(DateFr uneDate) { … } // Retourne true si la date est avant uneDate
    public boolean estApres(DateFr uneDate) { … } // Retourne true si la date est après uneDate
    public boolean estJourOuvrable() { … } // Retourne true si la date est un jour ouvrable
    public DateFr getJourSuivant() { … } // Retourne la date du jour suivant
}
```

11. Exemple d'utilisation d'une collection

```
Salarie salarie = new Salarie("D0001","LEGRAND","Paul") ;

List<Salarie> lesSalaries ; // Déclaration d'une collection d'instances de la classe Salarie
lesSalaries = new ArrayList<Salarie>() ; // Instanciation d'une collection

lesSalaries.add( salarie ) ; // Ajout d'un salarié
System.out.println( lesSalaries.get( 0 ) ) ; // Affichage du premier élément
System.out.println( lesSalaries.size() ) ; // Affichage du nombre d'éléments

for( Salarie unSalarie : lesSalaries ){ // Parcours de la collection (foreach dans d'autres langages)
        System.out.println(unSalarie.getNom()) ; // Affichage de l'élément courant
}
```

12. Extraits de pages internet concernant la dématérialisation

http://www.developpement-durable.gouv.fr : Ministère de l'écologie, du développement durable et de l'énergie
L'article 4 de l'arrêté du 09/11/1999 relatif aux documents de transport ou de location devant se trouver à bord des véhicules de transport routier de marchandises a été modifié. La lettre de voiture pourra être établie sous forme électronique.

http://www.usinenouvelle.com : TRANSWIDE DÉMATÉRIALISE LES DOCUMENTS DE TRANSPORTS
Les chargements n'arriveront pas plus vite, mais au moins tout le monde sera prévenu des retards. Après s'être attaquée à la dématérialisation des ordres de commandes, la jeune société irlandaise Transwide commercialise depuis quelques semaines un logiciel de gestion en ligne des documents de transport […]. Ce nouvel outil permet d'avoir accès en permanence aux documents remplis par les chauffeurs et les clients à chaque étape du transport. Concrètement, l'outil " twDoc " de Transwide dématérialise la lettre de voiture. La lettre de voiture met souvent plusieurs jours pour revenir à chacun de ces acteurs. Avec twDoc, ce document est créé de façon électronique sur les serveurs de Transwide. Chaque acteur de la chaîne de transport enregistre alors instantanément les informations qu'il doit légalement mentionner à chaque étape de l'expédition via un PC, un assistant numérique (PDA) ou un téléphone mobile GPRS. La signature se fait électroniquement grâce à un code d'accès avec mot de passe ou une infrastructure à clé publique (PKI). Les avantages sont multiples pour le transporteur. Au lieu d'attendre qu'on lui renvoie sa copie de la lettre de voiture, il dispose d'une vision immédiate de l'état de ses expéditions. Cela facilite la lecture et l'intégration de documents souvent remplis à la main de façon illisible. Si on fait le choix d'une liaison EDI ou XML, l'ensemble des données est automatiquement intégré au progiciel de gestion intégré de l'entreprise.

http://www.journaldunet.com
Pour qu'un projet de dématérialisation ne soit pas synonyme de fuite d'informations (et les exemples sont nombreux ces derniers temps : hacking de Bercy, piratage de comptes chez Sony...), il est nécessaire de mettre en place des règles et des politiques de sécurité physique et informatique permettant de garantir la sécurité des données. **Protéger l'information consiste à garantir sa disponibilité, son intégrité, sa confidentialité et sa traçabilité**.

http://www.businessmarches.com
La numérisation des documents et l'essor des communications en ligne accroissent le risque de pertes de données pour les entreprises.
L'usage d'ordinateurs zombies et de programmes destinés à pénétrer dans un système d'information ainsi que les erreurs causées par des tiers (services d'externalisation des données ou d'infogérance) sont incriminés. Si le fonctionnement des machines est notamment en cause, celui des applications est aussi en question : des logiciels endommagés ou fonctionnant mal peuvent engendrer l'illisibilité des fichiers ou affecter leur intégrité.

13. Scénario de chargement

Numéro de dossier : 1917 Phase : Chargement

1. Le véhicule immatriculé EZ-786-WL arrive sur le lieu de chargement le 5 avril 2016 à 8h45.
2. Le véhicule immatriculé ET-565-XQ arrive sur le lieu de chargement le 5 avril 2016 à 9h00.
3. Les deux véhicules quittent le lieu de chargement le même jour à 12h30.

14. Données échangées au format *Json*

Json (JavaScript Object Notation) est un format de données qui permet de représenter l'information à l'aide de paires clef/valeur, de listes ordonnées et d'objets.
Type MIME d'un document *Json* : ***application/json*** (**valeur à affecter à l'entête HTTP *Content-Type***)

Exemple :

```
{
      "dossier" : "1917",
      "phase" : "chargement",
      "vehicule" : [
            {
                  "immatriculation" : "EZ-786-WL",
                  "arrivee" : "2016-04-05 08:45:00",
                  "depart" : "2016-04-05 12:30:00"
            },
            {
                  "immatriculation" : "ET-565-XQ",
                  "arrivee" : "2016-04-05 09:00:00",
                  "depart" : "2016-04-05 12:30:00"
            }
      ]
}
```

15. Architecture *Rest*

Rest (REpresentational State Transfert) est un style d'architecture particulièrement bien adaptée aux services *web*. *Rest* identifie les ressources par des *URI* (Uniform Resource Identifier). Il s'appuie sur les méthodes du protocole *HTTP* pour exprimer les opérations et sur les codes de statut *HTTP* pour exprimer les résultats des opérations.

CRUD	Méthode *HTTP*	Code de statut
Create	POST	201 : Ressource créée, 409 : Ressource existante (conflit)
Read	GET	200 : Ressource(s) trouvée(s), 404 : Ressource non trouvée
Update	PUT	200 : Ressource modifiée, 404 : Ressource non trouvée
Delete	DELETE	200 : Ressource supprimée, 404 : Ressource non trouvée

Exemples :
GET /dossiers/ *Obtenir la représentation de tous les dossiers*
GET /dossiers/1917 *Obtenir la représentation du dossier numéro 1917*
POST /dossiers/ *Créer un nouveau dossier (représentation du nouveau dossier dans le corps de la requête). L'entête HTTP 'LOCATION' de la réponse a pour valeur l'URL de la ressource créée.*

16. Fichiers de l'application serveur (extraits)

Fichier *RealisationController.php*

L'extrait ci-dessous correspond au contrôleur « Realisation » associé au cas d'utilisation de la réalisation d'un chargement.

```
class RealisationController extends Controller {
        public function getFicheInstructionAction($numDossier){
        // Cette fonction est appelée par la méthode HTTP GET et retourne la fiche au format Json
                …
                $ficheInstruction = $dossier->getFicheInstruction() ;
                // Á cet endroit l'objet $ficheInstruction de la classe FicheInstruction
                // représente la fiche d'instruction du dossier $numDossier, si la fiche
                //d'instruction n'existe pas, $ficheInstruction a pour valeur null
                $reponse = new Response() ;
                // reste à coder :
                // test de l'existence de la fiche, préparation de l'entête HTTP
                // et envoi de la fiche au format Json
                return $reponse ;
        }
        public function chargementAction($numDossier){
        // Cette fonction est appelée par la méthode HTTP POST et enregistre les informations
        // du chargement dans la base de données
                ...
                // Création et enregistrement dans la base de données d'un objet de la classe
                // Chargement. En cas de succès, $idChargement a pour valeur
                // l'identifiant de l'objet créé. Sinon, cette variable a pour valeur false.
                // L'entête HTTP 'LOCATION' redirige vers la page de description du dossier créé.
                $reponse = new Response() ;
                if( $idChargement != false ){
                        // preparation de l'entête HTTP 'LOCATION'
                        $location = $this->get('request')->getSchemeAndHttpHost()
                                        . '/dossiers/'. $numDossier. '/chargement/' . $idChargement ;
                        $reponse->headers->set('Location',$location) ; // envoi de l'entête dans la réponse
                        $reponse->setContent('Redirection'); // ajout d'un contenu à la page
                        $reponse->setStatusCode(201) ;
                } else {
                        $reponse->setStatusCode(409) ;
                }
                return $reponse ;
        }

        public function livraisonAction(){
                …
                return $reponse ;
        }
        ...
}
```

Fichier *FicheInstruction.php* (extrait)

```
class FicheInstruction {
        private $numero ;
        private $dateCreation ;
        public function getNumero(){...}
        public function getDateCreation(){..}
        public function toJson(){...}    // Retourne l'objet au format Json
}
```