Le Sénat

CONCOURS D'INFORMATICIEN PROFIL « DÉVELOPPEUR »

SUJETS DONNÉS AU CONCOURS 2015-2016

IMPORTANT

Le programme étant toujours susceptible d'être modifié, cette brochure est fournie à titre purement indicatif.

Pour tout renseignement complémentaire concernant ce concours les candidats peuvent s'adresser à la :
Direction des Ressources humaines et de la Formation du Sénat
15, rue de Vaugirard – 75291 Paris cedex 06
Internet : http://www.senat.fr/emploi – Courriel : concours-rhf@senat.fr

SÉNAT

SOMMAIRE

ÉPREUVES ÉCRITES D'ADMISSIBILITÉ

1. Épreuve technique

Cette épreuve est destinée à tester les connaissances techniques et informatiques des candidats (culture informatique générale, programmation, logique, algorithmie).

Pour répondre aux questions de programmation, les candidats devront choisir parmi les langages suivants : C/C++, Java.

(durée 3 heures – coefficient 4)

2. Étude de cas

Les candidats doivent réaliser l'étude d'un projet applicatif comportant l'analyse du besoin, la conception, les choix techniques, le détail de la réalisation proposée (diagrammes pertinents en fonction de la méthode d'analyse et de conception choisie par le candidat, choix des modules, algorithmie).

Le dossier remis aux candidats pour cette épreuve pourra comporter des documents rédigés en anglais.

(durée 4 heures – coefficient 4)

ÉPREUVE TECHNIQUE

(Durée 3 heures – coefficient 4)

Le langage choisi devra être clairement indiqué au début de votre copie.
Il est interdit aux candidats de changer de langage en cours d'épreuve.
Le cas échéant, prenez le soin d'expliciter vos hypothèses.

Concernant les questions nécessitant l'écriture d'une « méthode », le code devra non seulement être correct mais aussi le plus lisible possible, c'est-à-dire correctement présenté (avec indentations et un minimum de rayures) et largement commenté. Les noms des fonctions et des variables devront également être choisis pour faciliter la compréhension du code. La lisibilité du code étant prise en compte dans la correction, il est vivement recommandé de préparer les réponses au brouillon.

Pour éviter une rédaction trop lourde lorsqu'il s'agit d'écrire une « méthode », il est inutile de réécrire la classe qui la contient.

Problème 1 : Questionnaire à choix multiple *(2 points – 0,2 point par question)*

Répondre à chaque question en indiquant le numéro de la question et la lettre ou les lettres correspondant à la bonne ou aux bonnes réponses. Pour chaque question, il existe au moins une bonne réponse.

S'agissant du barème de correction, aucun point ne sera retiré en cas de mauvaise réponse (pas de points « négatifs »). En revanche :

- aucun point ne sera attribué lorsque vous proposerez au moins une réponse erronée à une question ;
- lorsqu'il y a n réponses correctes à une question, chaque bonne réponse rapportera 0,2/n point. Le total pour la question est arrondi au dixième inférieur.

1 – Linus Torvald est un développeur à l'origine :

a) du noyau Linux

b) de git

c) de DOS

d) de Eclipse

e) de Fortran

2 – Le langage HTML a été inventé :

a) à l'INRIA

b) au CERN

c) au MIT

d) à Stanford

e) au DoD américain

3 – CSS est l'acronyme de :

a) Clear Short Styling

b) Coherent Style System

c) Complicated Style System

d) Cascading Style Sheet

e) CSS n'est pas un acronyme

4 – Quel(s) point(s) commun(s) y a-t-il entre les protocoles TLS, HTTP, DNS et TCP ?

a) Ils sont tous sur la 7ème couche (applicative) du modèle OSI

b) Ils sont encore utilisés aujourd'hui

c) Il n'en existe pas de version stable

d) Ils entrent tous en jeu lors d'une requête sur la page https://www.senat.fr/

e) Leur objet principal est l'échange de fichiers

5 – Laquelle (ou lesquelles) des propositions suivantes ne sont pas des codes HTTP corrects ?

a) ACK

b) 101

c) 200

d) 404

e) 601

6 – Le résultat de l'opération (2 & 3), en java ou C++, renvoie :

a) le booléen true

b) le booléen false

c) l'entier 2

d) l'entier 3

e) une erreur

7 – Le réusinage de code (refactoring), consiste en :

a) le recalcul de l'indentation d'un code

b) l'optimisation de l'occupation mémoire d'un programme

c) le portage d'un code sur un autre environnement

d) la réécriture d'un code dans un autre langage de programmation

e) la réorganisation d'un code dans le but d'en augmenter la lisibilité et la maintenabilité

8 – L'héritage multiple est le fait :

a) pour une classe d'hériter de plus d'un père

b) pour une classe d'avoir au moins un père et un grand-père

c) pour un code d'utiliser plusieurs bibliothèques logicielles

d) pour un programme d'être constitué de codes en différents langages

e) pour un code de reposer sur une machine virtuelle assurant la portabilité

9 – Un problème est de complexité NP :

a) peut être transformé avec certitude en un problème de complexité polynomiale

b) s'il n'existe pas une méthode exacte de résolution

c) si l’une de ses solutions potentielles peut être vérifiée en un temps polynomial

d) s'il ne peut être résolu simplement que sur des ordinateurs analogiques

e) s'il n'est pas possible de donner une évaluation de son temps de résolution

10 – Qu’est-ce qu’un Qubit ?

a) Un bit qui peut prendre 3 valeurs 0, 1 ou indéterminé

b) La plus petite unité d'information dans la logique floue

c) Un bit de parité servant à la détection/correction d'erreur

d) La plus petite unité d'information pour un ordinateur quantique

e) Pour un nombre donné, le bit non nul le plus significatif

Problème 2 : Qu'affiche ce programme ? *(3 points – 0,5 point par fonction et 1 point pour la fonction f3)*

Qu'affiche le programme suivant ? Il n'est pas demandé de justifier sa réponse mais simplement d'écrire ce qui est affiché.

Le candidat ne considérera que la partie rédigée dans le langage qu'il a choisi et indiqué au début de sa copie.

- **Java**

```
class Parent {

    int v = 20;

    public int f1() {
        return (v / 2) % 10;
    }

    public static void f2(int debut) {
        int compteur = debut;
        for (int i = 0; i < 100; i++) {
            if (compteur >= debut) {
                compteur += compteur;
            } else {
                compteur = 0;
            }
        }
        System.out.println("f2 : " + compteur);
    }

    public static boolean f3() {
        boolean[][] a = {{true, true, false}, {false, true, true}};
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < 6; j++) {
                a[i][j % 3] = a[j % 2][i % 3] && a[i][j % 3];
            }
        }
        return a[1][2] ? (a[0][0] && !a[1][0]) : (a[0][1] || a[0][2] ||
a[1][1]);
    }

    public static void f4() {
        int a = 1234567890;
        int b = 10;
        int c = 0;
        while (a > 0) {
            c *= b;
            c += a % b;
            a /= b;
        }
        System.out.println("f4 : " + c);
    }
    public static int f5(int a) {
        if( a <= 5 ) return a;
        if( (a%2) == 1 ) return f5(a*4);
        return f5(a/3);
    }
}

class A extends Parent {
```

```
    public A() {
        v = 30;
    }

    public int f1(int w) {
        return (w + v) / 3;
    }
}

class B extends A {

    public int f1() {
        return super.f1(super.f1()) + 2;
    }
}

public class QuelResultat {

    public static void main(String[] args) {
        System.out.println("f1 : "+(new B()).f1());
        Parent.f2(1000);
        System.out.println("f3 : " + Parent.f3());
        Parent.f4();
        System.out.println("f5 : " + Parent.f5(17));
    }
}
```

- **C++**

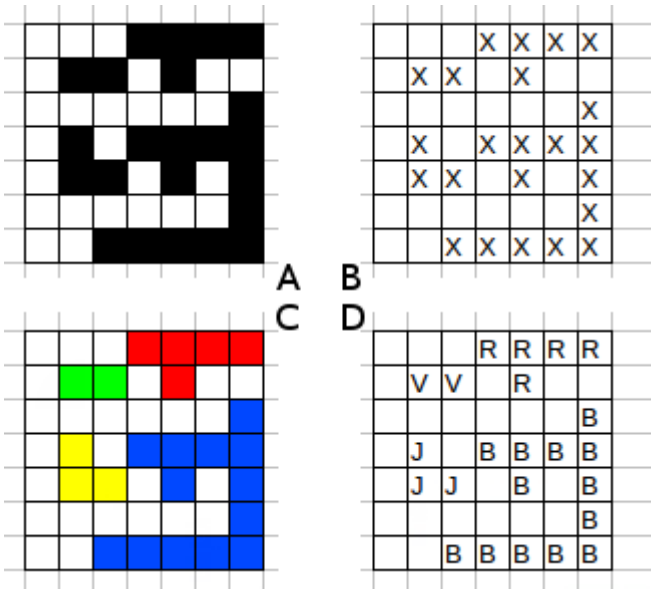
```
#include <iostream>

class Parent
{
protected:
  int v;

public:
  Parent() : v(10)
  {
  }

  int f1()
  {
     return (v / 2) % 10;
  }

  static void f2(int debut)
  {
     int compteur = debut;
     for (int i = 0; i < 100; i++)
        {
          if (compteur >= debut) compteur += compteur;
          else compteur = 0;
        }
     std::cout << "f2 : " << compteur << "\n";
  }

  static bool f3()
  {
     bool a[2][3] = {{true, true, false}, {false, true, true}};
     for (int i = 0; i < 2; i++)
        for (int j = 0; j < 6; j++)
          a[i][j % 3] = a[j % 2][i % 3] && a[i][j % 3];
     return a[1][2] ? (a[0][0] && !a[1][0]) : (a[0][1] || a[0][2] ||
```

```
a[1][1]);
  }

  static void f4()
  {
     int a = 1234567890;
     int b = 10;
     int c = 0;
     while (a > 0)
        {
          c *= b;
          c += a % b;
          a /= b;
        }
     std::cout << "f4 : " << c << "\n";
  }

  static int f5(int a)
  {
     if( a <= 5 ) return a;
     if( (a%2) == 1 ) return f5(a*4);
     return f5(a/3);
  }
};

class A : public Parent
{
public:
  A()
  {
     v = 30;
  }

  int f1(int w)
  {
     return (w + v) / 3;
  }
};

class B : public A
{
public:
  int f1()
  {
     return f1(Parent::f1()) + 2;
  }
};

int main( int argc, const char **argv)
{
  B b;
  std::cout << "f1 : " << b.f1() << "\n";
  Parent::f2(1000);
  std::cout << "f3 : " << (Parent::f3() ? "true" : "false") << "\n";
  Parent::f4();
  std::cout << "f5 : " << Parent::f5(17) << "\n";
}
```

Problème 3 : Coloriage de composantes connexes *(6 points)*

Le but de cet exercice est de réaliser un programme pour colorier, avec des couleurs distinctes, les composantes connexes d'une image en noir et blanc. Une composante connexe est formée de pixels de même couleur et voisins de proche en proche. Deux pixels sont voisins lorsqu'ils possèdent un bord en commun. Par exemple, dans l'image ci-dessous (**A**), il y a une composante connexe de couleur blanche et quatre composantes connexes de couleur noire. Un coloriage correspondant pourrait être celui retrouvé en **C**. Pour plus de simplicité, on ne se préoccupera que d'une représentation sous forme de caractères (mode terminal). On retrouve donc en **B** la représentation de l'image en noir et blanc et en **D** le coloriage des composantes connexes.

Le candidat ne considérera que la partie rédigée dans le langage qu'il a choisi et indiqué au début de sa copie.



Dans la suite de ce problème, les images en noir et blanc sont représentées par des matrices carrées de booléens (la valeur `true` représente un pixel noir). Le code suivant est proposé :

- **Java**

```
import java.util.Hashtable;
import java.util.Random;

class Uf {

    private int[] rang;
    private int[] parent;

    Uf(int n) {
        rang = new int[n];
        parent = new int[n];
```

```
        for (int i = 0; i < n; i++) {
            parent[i] = i;
        }
    }

    // TODO QUESTION 2
    int find(int i) {
        int p = parent[i];
        if (p == i) {
            return (i);
        } else {
            int r = find(p);
            return (r);
        }
    }

    void link(int i, int j) {
        parent[j] = i;
    }

    void union(int i, int j) {
        int ri = find(i);
        int rj = find(j);
        if (ri != rj) {
            if (rang[ri] < rang[rj]) {
                parent[ri] = rj;
            } else {
                parent[rj] = ri;
                if (rang[ri] == rang[rj]) {
                    rang[ri]++;
                }
            }
        }
    }
}



abstract class Image {

    boolean[][] t;
    private Random gen;
    int size;

    Image(int n) {
        size = n;
        gen = new Random();
        t = new boolean[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                t[i][j] = gen.nextBoolean();
            }
        }
    }

    int toInt(int i, int j) {
        return (i * size + j);
    }

    abstract void dessineNB();

    abstract void dessineCouleur(Uf uf);
}
```

```
class ImageTerminal extends Image {

    ImageTerminal(int n) {
        super(n);
    }

    void dessineNB() {
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                if (t[i][j]) {
                    System.out.print("X");
                } else {
                    System.out.print(" ");
                }
            };
            System.out.println("");
        }
        System.out.println("\n\n");
    }

    void dessineCouleur(Uf uf) {
      // TODO QUESTION 3
    }

    /**
     * Renvoie une nouvelle couleur à chaque nouvel appel. Le recyclage des
     * couleurs si le nombre d'appels dépasse le nombre de caractères ASCII
     * disponible est géré par la méthode, il n'est donc pas demandé au
candidat
     * de gérer les cas aux limites.
     *
     * @return un caractère représentant une couleur.
     */
    private char getNouvelleCouleur() {
        // L'implémentation n'est pas demandée pour la question 3
    }
}

public class Coloriage {

    static Uf calculComposantesConnexes(Image img) {
        Uf uf = new Uf(img.size * img.size);

        for (int i = 0; i < img.size; i++) {
            for (int j = 0; j < img.size; j++) {
                if (i < img.size - 1) {
                    if (img.t[i][j] == img.t[i + 1][j]) {
                        // TODO QUESTION 1
                        uf.link(img.toInt(i, j), img.toInt(i + 1, j));
                    }
                }
                if (j < img.size - 1) {
                    if (img.t[i][j] == img.t[i][j + 1]) {
                        // TODO QUESTION 1
                        uf.link(img.toInt(i, j), img.toInt(i, j + 1));
                    }
                }
            }
        };
        return (uf);
    }

    /**
     * Lance le programme sur une image carrée de 10 pixels de côté.
     *
     * @param args aucun argument n'est attendu en entrée.
```

```
     */
    public static void main(String[] args) {
        Image img = new ImageTerminal(10);
        img.dessineNB();
        Uf uf = calculComposantesConnexes(img);
        img.dessineCouleur(uf);
    }
}
```

- **C++**

```
#include <stdlib.h>
#include <stdio.h>

class Uf;


class Image
{
public:
  int size;
  bool **t;

  Image(int n) : size(n), t(0)
  {
     srand(time(0)); // initialisation de la série pseudo-aléatoire
     t = new bool *[n];
     for( int i = 0; i < n; i++ ) t[i] = new bool[n];
     for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
             t[i][j] = (random() % 2 == 0);
  }

  ~Image()
  {
     if( t )
        {
          for( int i = 0; i < size; i++ ) delete(t[i]);
          delete(t);
        }
  }

  inline int toInt(int i, int j) const
  {
     return (i * size + j);
  }

  virtual void dessineNB() = 0;

  virtual void dessineCouleur(Uf &uf) = 0;
};


class Uf
{
private:
  int *rang;
  int *parent;

public:
  Uf(const Image &img) : rang(0), parent(0)
  {
     int n = img.size * img.size;
     rang = new int[n];
     parent = new int [n];
```

```
      for (int i = 0; i < n; i++) {
         parent[i] = i; rang[i] = 0;
      }
   }

   ~Uf()
   {
      if( parent ) delete parent;
      if( rang ) delete rang;
   }

   // TODO QUESTION 2
   int find(int i)
   {
      int p = parent[i];
      if (p == i) return i;
      else return find(p);
   }

   void link(int i, int j)
   {
      parent[j] = i;
   }

   void union_t(int i, int j)
   {
      int ri = find(i);
      int rj = find(j);
      if (ri != rj)
         {
           if (rang[ri] < rang[rj])
              parent[ri] = rj;
           else
              {
                 parent[rj] = ri;
                 if (rang[ri] == rang[rj])
                   rang[ri]++;
              }
         }
   }
};


class ImageTerminal : public  Image
{
private:

public:
   ImageTerminal(int n) : Image(n) {}

   void dessineNB()
   {
      for (int i = 0; i < size; i++)
         {
           for (int j = 0; j < size; j++)
              {
                 if (t[i][j])
                   putchar('X');
                 else
                   putchar(' ');
              }
           putchar('\n');;
         }
      puts("\n");
   }

   void dessineCouleur(Uf &uf)
```

```
  {
      // TODO QUESTION 3
  }

private:
 /**
  * Renvoie une nouvelle couleur à chaque nouvel appel. Le recyclage des
  * couleurs si le nombre d'appels dépasse le nombre de caractères ASCII
  * disponible est géré par la méthode, il n'est donc pas demandé au candidat
  * de gérer les cas aux limites.
  *
  * @return un caractère représentant une couleur.
  */
  char getNouvelleCouleur()
  {
     // TODO QUESTION 3
  }
};


class Coloriage
{
public:
  void calculComposantsConnexes(Uf &uf, const Image &img)
  {
     for (int i = 0; i < img.size; i++)
        {
          for (int j = 0; j < img.size; j++)
             {
             if (i < img.size - 1)
                {
                  if (img.t[i][j] == img.t[i + 1][j]) {
                     // TODO QUESTION 1
                     uf.link(img.toInt(i, j), img.toInt(i + 1, j));
                  }
                }
             if (j < img.size - 1)
                {
                  if (img.t[i][j] == img.t[i][j + 1])
                     {
                     // TODO QUESTION 1
                     uf.link(img.toInt(i, j), img.toInt(i, j + 1));
                     }
                }
             }
        };
  }


  void run(Image &img)
  {
     img.dessineNB();
     Uf uf(img); calculComposantsConnexes(uf, img);
     img.dessineCouleur(uf);
  }

public:

 /**
  * Lance le programme sur une image carrée de 10 pixels de côté.
  *
  * @param args aucun argument n'est attendu en entrée.
  */
  void main()
  {
     ImageTerminal img(10);
     run(img);
```

```
  }
};

int main(int argc, const char** argv)
{
  Coloriage coloriage; coloriage.main();
  return 0;
}
```

- **Questions**

1°) Expliquez pourquoi l'algorithme implémenté ne fonctionne pas dans tous les cas et serait corrigé en remplaçant les deux appels `uf.link` par des appels `uf.union` avec les même paramètres (vous trouverez ces appels en dessous des commentaires `TODO QUESTION 1`). ***(2,5 points)***

2°) À quoi sert le tableau rang ? ***(1 point)***

3°) Proposez, dans le langage choisi, une modification très simple sur le code de la méthode `find` de la classe `Uf` qui permette de limiter les parcours d'arbres. ***(1 point)***

4°) Implémentez dans le langage choisi la méthode `dessineCouleur` de la classe `ImageTerminal` en utilisant la méthode `getNouvelleCouleur` qui renvoie une nouvelle couleur à chaque appel. On considérera que cette méthode gère le recyclage de couleurs lorsque le nombre de composantes connexes dépasse le nombre des caractères possibles sur le terminal. À toutes fins utiles, on suppose que l'on dispose d'une classe Hachage qui implémente un tableau associatif permettant d'associer un entier à un caractère. Les méthodes de cette classe sont :

- `void inserer(int cle, char valeur)` => associe la valeur `valeur` à la clé `cle`
- `void estDefini(int cle)` => retourne `true` si une valeur est associée à la clé `cle`
- `char getValeur(int cle)` => retourne la valeur associée à la clé `cle`

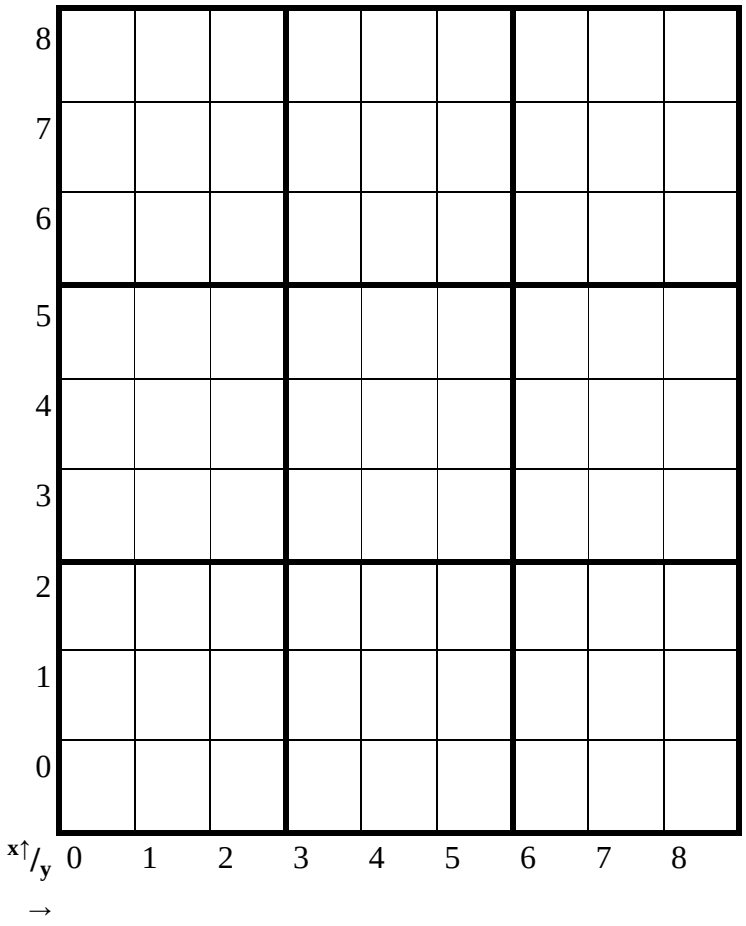
(1,5 point)

Problème 4 : Sudoku *(9 points)*

Dans ce problème, la question 1 doit nécessairement être traitée pour pouvoir traiter les questions suivantes. Les questions 3 et 4 d'une part, et les questions 5, 6, 7 et 8 d'autre part, sont liées. Toutefois, en cas de difficulté avec une question, il est possible de traiter la question suivante, moyennant quelques hypothèses, notamment concernant les signatures des méthodes qui auraient dû être écrites précédemment.

Le candidat traitera le problème dans le langage qu'il a choisi et indiqué au début de sa copie.

Une grille de Sudoku est un carré de neuf cases de côté, subdivisé en sous-grilles carrées 3x3 identiques appelées « **régions** » et délimitées par des lignes en gras dans le schéma ci-dessous.



Par convention (x,y) désigne la case se trouvant dans la ligne x et dans la colonne y.

Chaque case peut être vide ou contenir un nombre de 1 à 9. Les règles suivantes pour un nombre donné de 1 à 9 doivent être respectées :

(**i**) une ligne ne peut contenir qu'une seule fois ce nombre ;

(**ii**) une colonne ne peut contenir qu'une seule fois ce nombre ;

(**iii**) une région ne peut contenir qu'une seule fois ce nombre.

Le jeu de Sudoku consiste à partir d'une grille contenant déjà certains nombres respectant (i) à (iii), de la remplir progressivement afin d'obtenir *in fine* une grille totalement remplie respectant

(i) à (iii).

1°) Écrire une classe représentant une grille de Sudoku en donnant en français les précisions que vous jugerez utiles. ***(0,5 point)***

2°) Écrire la méthode initialiser() qui prend en argument une tableau 9x9 d'entier de 0 à 9 et qui remplit la structure de donnée mise au point dans la question 1. Dans le tableau d'entiers, 0 représente une case vide. ***(0,5 point)***

3°) Écrire une méthode verifier() qui prend en argument la position x, y d'une case de la grille de Sudoku et qui renvoie true si les règles (i) à (iii) sont vérifiées pour cette case, c'est-à-dire par exemple que la ligne x ne contient pas d'autres occurrence du nombre contenu dans la case (x,y), que la colonne y ne contient pas d'autres occurrences du nombre contenu dans la case (x,y), etc. ***(1 point)***

4°) Écrire la méthode verifier_tout() qui vérifie les règles (i) à (iii) pour l'ensemble de la grille en appelant la méthode verifier() de la question 3 pour chaque case de la grille. Quelle est la complexité de cette méthode ? Est-elle optimale ? Si oui, expliquez pourquoi. Si non, expliquez comment nous pourrions l'optimiser. ***(1 point)***

5°) On suppose que la grille contient déjà quelques cases remplies en respect des règles (i) à (iii). Écrire une méthode remplir() qui prend en argument un nombre n de 1 à 9 et applique l'algorithme suivant ***(3 points)*** :

1 – la méthode détermine quelles cases ne pourraient pas être remplies avec n. Par exemple, si la case (x,y) contient n, on peut en déduire d'après (i) qu'aucune case de la ligne x ne peut contenir n, d'après (ii), qu'aucune case de la colonne y ne peut contenir n, etc. ;

2 – une fois ce travail fait, si on constate que pour chaque règle (i) à (iii), il n'existe qu'une case pouvant contenir n, on remplit cette case avec n. Par exemple, par application de la règle 1, si on constate que sur la ligne x' seule la case (x',y') peut contenir n, alors on remplit (x',y') avec n ;

3 – si, en revanche, pour une des règles (i) à (iv), aucune case ne peut contenir n, la méthode détecte une erreur : il est impossible de remplir la grille complètement.

6°) Écrire une méthode remplir_tout() qui par application répétée de la méthode remplir() de la question 5, essaie de remplir le maximum de cases. ***(1 point)***

7°) Une fois la méthode remplir_tout() appelée, il peut rester des cases non remplies. On peut alors être tenté de remplir une case vide avec une certaine valeur (a) et d'appliquer à nouveau remplir_tout() de la question 6 pour voir si on arrive à remplir plus de cases, ou bien si on arrive à une contradiction. Expliquez de manière concise comment on pourrait procéder ainsi par essais successifs afin de remplir totalement la grille. Des structures de données supplémentaires sont-elles nécessaires ? Quels sont les éventuels problèmes que pourrait poser l'exécution de l'algorithme ? ***(1 point)***

8°) Selon vous, le choix de la case vide en (a) a-t-il une importance sur l'efficacité de l'algorithme ? Expliquez. ***(1 point)***

ÉTUDE DE CAS

(Durée 4 heures – coefficient 4)

Introduction

La Direction de l'Accueil et de la Sécurité (DAS) est composée de deux services : le Service de surveillance et de contrôle qui regroupe les 80 surveillants du Palais et le Service de surveillance du Jardin qui regroupe les 30 surveillants du Jardin du Luxembourg.

Les surveillants du Palais bénéficient, sous certaines conditions, du versement de primes forfaitaires et d'heures supplémentaires calculées aujourd'hui à la main. La DAS souhaite automatiser le calcul de ces primes.

Vous êtes l'informaticien en charge de l'étude et du développement de cette automatisation.

Vous trouverez ci-après :

- une description de l'existant :
  - le processus actuel,
  - l'environnement applicatif,
  - la règlementation applicable pour le calcul des primes ;
- les objectifs du projet ;
- la liste des questions auxquelles vous devez répondre.

Description de l'existant

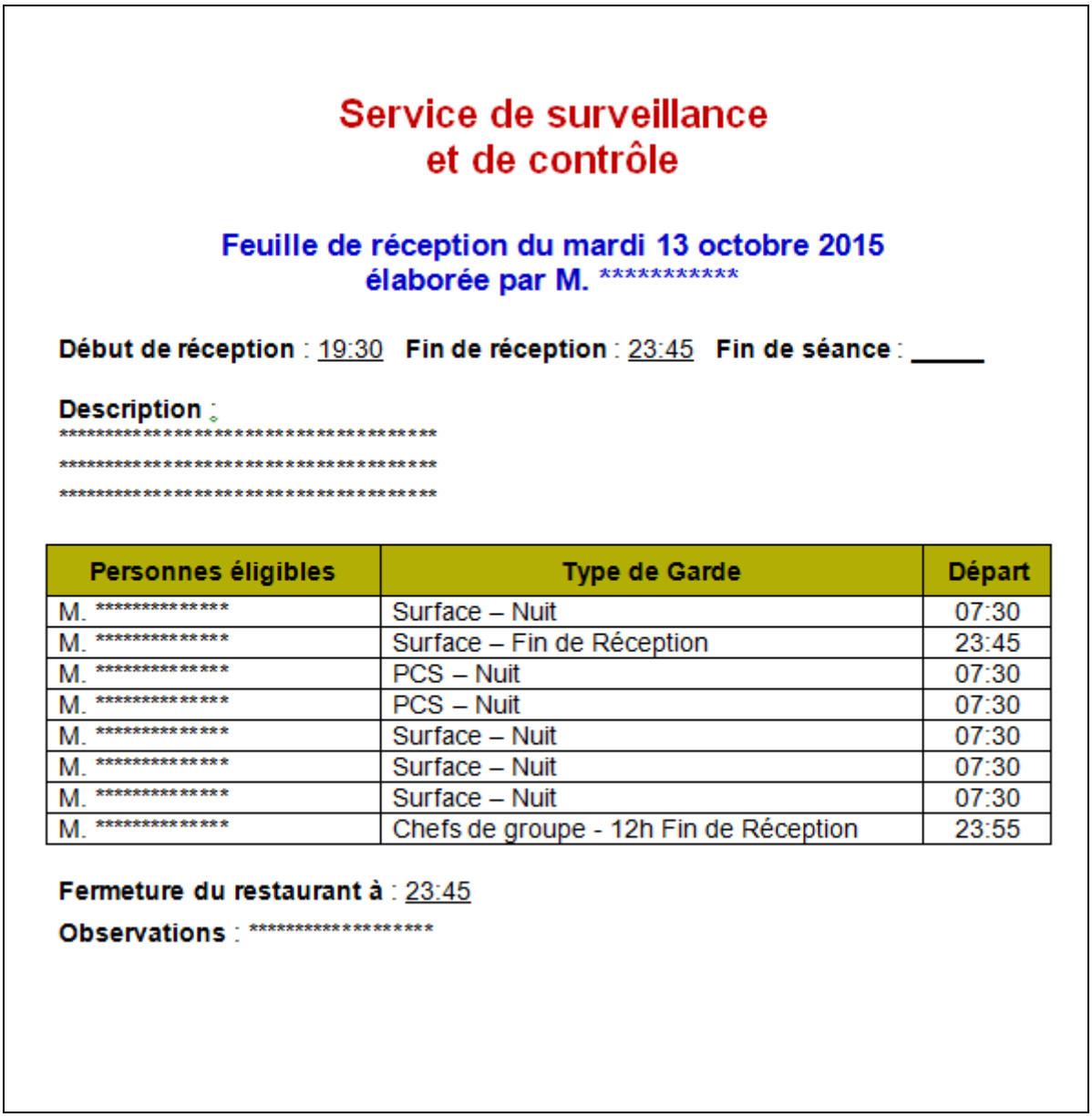
Le processus actuel

Les surveillants du Palais perçoivent, sous certaines conditions, des primes forfaitaires et des heures supplémentaires pour le travail effectué le dimanche et les jours fériés, ou en cas de réception occasionnant une charge de travail supplémentaire.

Les surveillants du Palais sont surveillant de surface, contrôleur du PCS (PC sécurité) ou chef de groupe. Les surveillants de surface et contrôleurs sont encadrés par les chefs de groupe. Les horaires de travail des surveillants du Palais et le type de garde qu'ils effectuent sont saisis par les chefs de groupe dans l'application Chronos (voir chapitre suivant). Les chefs de groupe effectuent également des gardes et saisissent leurs propres horaires.

Pour calculer et valider les primes, on utilise un processus papier. Une feuille de réception est élaborée par un chef de groupe du Service de surveillance et de contrôle indiquant une description de la réception, des observations éventuelles, l'heure de début et de fin de la réception, l'heure de fin de séance publique, l'heure de fermeture du restaurant ainsi que la liste des surveillants éligibles à la prime.

Les surveillants du Palais ne font jamais plus d'une réception par jour.



Service de surveillance et de contrôle

Feuille de réception du mardi 13 octobre 2015 élaborée par M. **********

Début de réception : 19:30 **Fin de réception** : 23:45 **Fin de séance** : _____

Description :
****
****
****

Personnes éligibles	Type de Garde	Départ
M. **********	Surface – Nuit	07:30
M. **********	Surface – Fin de Réception	23:45
M. **********	PCS – Nuit	07:30
M. **********	PCS – Nuit	07:30
M. **********	Surface – Nuit	07:30
M. **********	Surface – Nuit	07:30
M. **********	Surface – Nuit	07:30
M. **********	Chefs de groupe - 12h Fin de Réception	23:55

Fermeture du restaurant à : 23:45

Observations : **********

Figure 1 : Exemple de feuille de réception

Après vérification par le chef du Service de surveillance et de contrôle de la DAS (le supérieur hiérarchique des chefs de groupe), ces feuilles sont traitées mensuellement par un secrétaire administratif de la DAS pour déterminer les droits à primes ; cela donne lieu à un récapitulatif.

Service de surveillance et de contrôle

Primes de réception et pour service du dimanche et des jours fériés Cadre des Surveillants du Palais

Mois : janvier 2015

Nom - Prénom	Matricule	Primes de réception						H. sup	Dim. et Fériés
		jusqu'à 20 h		après 20 h		après 23 h			
		Resp. Taux 1	Autres Taux 1	Resp. Taux 2	Autres Taux 2	Resp. Taux 3	Autres Taux 3		
M. **********	**********	0	0	0	1	0	0	0,0	0,0
M. **********	**********	0	0	1	0	0	0	0,5	0,0
M. **********	**********	0	0	0	0	0	1	0,0	0,0
M. **********	**********	0	0	0	1	0	1	1,0	0,0
M. **********	**********	0	0	0	1	0	0	0,0	0,0
M. **********	**********	0	0	0	1	0	1	0,0	0,0
M. **********	**********	0	0	0	0	0	1	0,0	0,0
...	...	...	...	...	...	...	...	...	...
Récapitulatif		**0**	**0**	**9**	**26**	**0**	**40**	**10,5**	**4,0**

Le directeur de l'accueil et de la sécurité

Figure 2 : Exemple de récapitulatif

Après signature du récapitulatif par le directeur, celui-ci est saisi dans le progiciel de paye HR Access par un fonctionnaire de la Direction Financière.

L'état récapitulatif contient uniquement un dénombrement des primes et des heures supplémentaires. Ces éléments sont transformés en montants uniquement dans le progiciel de paye.

L'environnement applicatif

Annuaire LDAP

Un annuaire LDAP centralisé stocke, entre autres, les informations administratives des personnels et des organisations (directions, services, divisions, etc.). Chaque fonctionnaire est identifié par un matricule unique du type PER/XXXXXY où X sont des chiffres et Y une lettre.

Les surveillants du Palais se distinguent des autres personnels par leur **cadre** d'emplois. Un cadre d'emplois regroupe les fonctionnaires soumis aux mêmes règles et occupant des fonctions similaires ou de même « niveau » (exemple de cadres d'emplois : cadre des surveillants du Palais, cadre des surveillants du Jardin, cadre des administrateurs, cadre des administrateur-adjoints, cadre des secrétaires administratifs, etc.).

La notion **d'unité organisationnelle** permet de regrouper les surveillants du Palais selon leurs fonctions, à savoir : contrôleurs du PCS, surveillants de surface, chefs de groupe ou chef de service.

Le **poste** occupé par une personne est identifié par un code. À une date donnée, une personne occupe un seul poste et un poste n'est occupé que par une seule personne. La Direction des Ressources humaines actualise au fil de l'eau l'affectation des personnels sur les postes dans un logiciel de ressources humaines qui fait l'objet d'une synchronisation quotidienne avec l'annuaire LDAP. Ce dernier ne stocke que la **situation courante** des personnels et ne stocke pas l'historique des postes occupés.

Chaque individu a un login et un mot de passe géré par le LDAP. Les applications internes interrogent le LDAP pour vérifier le mot de passe des utilisateurs.

Les applications internes du Sénat peuvent interroger le LDAP pour récupérer les informations concernant les personnels et en particulier :

- nom ;
- prénom ;
- matricule ;
- login ;
- cadre ;
- direction ;
- unité organisationnelle ;
- poste.

Chronos

Les horaires des personnels du Sénat sont enregistrés dans une application maison appelée *Chronos*. La journée de travail est découpée en trois parties séparées par la pause du déjeuner et la pause du dîner. S'agissant du cadre des surveillants du Palais, le type de garde effectué par chacun d'entre eux est enregistré en plus des horaires du matin, de l'après-midi et du soir. Le logiciel Chronos est une application Web utilisée en Intranet. Elle est écrite en Java, utilise la technologie JSP (Java Server Pages) et une base de données ORACLE.

Les tables simplifiées sont les suivantes :

Personne					
Matricule	Nom	Prénom	Cadre	Unité Organisationnelle	Poste
			- Surveillants du Palais - Surveillants du Jardin - …	- Contrôleurs - Surveillants de surface - Chefs de groupe du Service de surveillance et de contrôle - Chef du Service de surveillance et de contrôle - …	

Relevé quotidien								
Date	Matricule	Début matin	Fin matin	Début après-midi	Fin après-midi	Début soir	Fin soir	ID type de garde

Astreinte					
Date	Matricule	Heure Début	Heure Fin	Heure Début déclenchement	Heure Fin déclenchement

Type de Garde		
ID	Libellé	Catégorie
	Chefs de groupe – Jour	J
	Chefs de groupe – 12h Fin de réception	FR
	Chefs de groupe – Fin de réception	FR
	Surface – Jour	J
	Surface – Nuit	N
	Surface – Fin de réception	FR
	PCS – Jour	J
	PCS – Nuit	N
	PCS – Fin de réception	FR

Les types de garde peuvent être regroupés en trois grandes catégories :

- jour (J) ;
- nuit (N) ;
- fin de réception (FR).

Lorsqu'un surveillant du Palais est de « fin de réception », son heure de départ n'est pas connue à l'avance. Il reste alors en service jusqu'à la fin de la séance publique ou jusqu'à la fin d'une réception (l'heure la plus tardive des deux).

Logiciel de paye

La paye est calculée par le progiciel HR Access. Les éléments nécessaires à l'intégration des primes dans le calcul de paye sont regroupés dans une table dont voici une vue simplifiée :

Éléments de prime									
Mois	Matricule	Primes réception taux 1 responsables	Primes réception taux 1 autres	Primes réception taux 2 responsables	Primes réception taux 2 autres	Primes réception taux 3 responsables	Primes réception taux 3 autres	Heures supplémentaires	Service du Dimanche

La DAS n'a pas accès au progiciel HR Access ; seule la Direction Financière y a accès.

La réglementation (note interne d'un administrateur de la DAS)

Modalités de versement des primes de réception

Les surveillants du Palais peuvent prétendre au bénéfice de la prime de réception dès lors qu'une **réception** (repas au restaurant, réunion, etc.) **organisée dans le Palais ou ses dépendances occasionne une charge de travail supplémentaire**.

Remarque préalable : dans l'énoncé suivant, il faut distinguer les primes aux taux 1, 2 ou 3 réservées aux chefs de groupe des primes aux taux 1, 2 ou 3 accordées aux autres surveillants. Cette distinction apparaît clairement dans l'état récapitulatif.

- **Cas n° 1 : pas de séance publique ou levée de la séance publique avant 19 heures**

Les surveillants du Palais qui sont de service peuvent prétendre au bénéfice de la prime de réception dès lors que la **fin de la réception intervient après 20 heures**.

- Si la réception se termine **avant 23 heures**, les surveillants de nuit et de fin de réception (FR) ainsi que le (ou les) chef(s) de groupe présent(s) bénéficient d'**une prime de réception au taux 2**.

- Si la réception se termine **après 23 heures** :

  - les **surveillants de nuit** bénéficient d'**une prime de réception au taux 3** ;

  - les **surveillants FR** et les **chefs de groupe** bénéficient d'une **prime de réception au taux 2** ainsi que d'**heures supplémentaires**. Ces heures supplémentaires sont comptabilisées par demi-heure à partir de 23 heures (toute demi-heure entamée étant due).

- **<u>Cas n° 2 : levée de la séance publique après 19 heures</u>**

Les surveillants du Palais qui sont de service peuvent prétendre au bénéfice de la prime de réception dès lors que la **fin de la réception intervient après 23 heures** et qu'un **délai de trois heures** s'écoule entre la levée de la séance publique et la fin de réception. Si ces deux conditions sont réunies :

- les **surveillants de nuit** bénéficient d'**une prime de réception au taux 3** ;

- les **surveillants FR** et les **chefs de groupe** bénéficient d'une **prime de réception au taux 2** ainsi que d'**heures supplémentaires**. Ces heures supplémentaires sont comptabilisées par demi-heure à partir de 23 heures (toute demi-heure entamée étant due).

Prime du dimanche et des jours fériés

Les astreintes effectuées le dimanche et les jours fériés par les chefs de groupe des surveillants du Palais donnent lieu au versement d'**une prime « pour service du dimanche et des jours fériés »**. Cette prime est majorée de 50 % lorsque l'astreinte a comporté une intervention justifiant une présence effective au Palais (c'est-à-dire une « astreinte déclenchée »).

Par ailleurs, dans le cas où une réception est organisée au Palais un dimanche ou un jour férié **alors qu'il n'y a pas de séance publique**, tous les surveillants du Palais de service (y compris les chefs de groupe) peuvent prétendre au bénéfice d'une **prime de réception** selon les modalités suivantes :

- versement au taux 1 si la réception se termine avant 20 heures ;

- versement au taux 2 si la réception se termine avant 23 heures ;

- versement au taux 3 si la réception se termine après 23 heures.

Objectifs

Les objectifs sont les suivants :

1) automatiser le calcul des primes (réalisé jusqu'alors par un secrétaire administratif de la DAS) et la production des documents papier par une extension de l'application Intranet Chronos ;

2) conserver un circuit de validation papier pour les feuilles de réception et les récapitulatifs de primes ;

3) permettre à un administrateur-adjoint de la DAS de produire, à partir de Chronos, des statistiques annuelles sur le nombre de primes accordées ;

4) éviter une saisie du récapitulatif des primes dans le logiciel de paye.

Questions

Prenez soin de justifier vos réponses et d'expliciter vos hypothèses. Il sera tenu compte dans la notation de la propreté de votre copie et de la clarté de vos réponses.

Vous avez mené un premier entretien avec la DAS et la Direction Financière au cours duquel le processus actuel vous a été décrit et une note sur la règlementation vous a été remise, ainsi que des exemples de documents au format papier.

1) ***Gestion de projet (1 point)***
Quelles tâches vous semblent utile d'organiser dans l'immédiat ? Vous listerez ensuite les grandes étapes qui vous permettront de mener ce projet à son terme.

2) ***Gestion des risques (1 point)***
Quels risque(s) ou difficulté(s) envisagez-vous *a priori* pour ce projet ? Comment les atténuer ?

3) ***Compréhension du besoin (3 points)***
Les horaires Chronos suivants ont été enregistrés pour les journées des jeudi 7 et vendredi 8 mai 2015 :

Prénom	Nom	Matricule	Date	Type de garde	Début matin	Fin matin	Début après-midi	Fin après-midi	Début soir	Fin soir
Pierre	DUPONT	PER/00001A	07/05/2015	Surface-Nuit					19 :30	07 :30
Paul	DURAND	PER/00002B	07/05/2015	Surface - Fin de réception	07 :30	12 :30	13 :30	19 :30	20 :00	00 :05
Jeanine	DUVAL	PER/00003C	07/05/2015	Chef de Groupe – Fin de réception	09 :00	12 :00	13 :00	18 :30	19 :00	00 :50
Éric	MARTIN	PER/00004D	08/05/2015	Surface - Jour	07 :30	12 :30	12 :30	19 :30		

Le 7 mai 2015, la séance publique s'est terminée à 19h30, et une réception a eu lieu de 20h00 à 23h55.

Le 8 mai, le chef de groupe Jérôme DESCHAMPS (matricule PER/12345E) était d'astreinte à son domicile. Toutefois, son astreinte n'a pas été déclenchée.

Dressez le récapitulatif des primes correspondant à ces deux journées.

4) ***Processus (1,5 point)***
Présentez les **acteurs** du nouveau système et proposez un **workflow** d'utilisation dans le formalisme de votre choix.

5) ***Modélisation (2 points)***

Proposez un **modèle objet** en utilisant, de préférence, un diagramme de classes au formalisme UML (sans indiquer le type des attributs). Si vous utilisez un autre formalisme, vous devrez indiquer celui que vous avez choisi.

6) ***Persistance des données (2 points)***

Proposez un modèle physique de données de **type relationnel**. Précisez les tables, attributs (sans les types) et contraintes que vous envisagez.

7) ***Architecture (2 points)***

D'une manière générale, présentez les choix d'architecture possibles permettant de faire cohabiter une approche objet avec un stockage dans une base relationnelle. Justifiez les avantages et les inconvénients des différentes architectures possibles.

8) ***Autorisations (1 point)***

Cette application s'inscrit dans le système d'information du Sénat qui est utilisé par 1200 personnes. Ces personnes sont soumises à mobilité (changement de poste au sein de l'administration du Sénat) ; il y a en moyenne une vingtaine de changements de poste par mois. Présentez un moyen de gérer des niveaux d'autorisations différents pour les différents utilisateurs.

9) ***Sécurité (2 points)***

Contre quels problèmes d'injection devez-vous vous prémunir ? Comment ?

10) ***Intégration (1 point)***

Expliquez comment vous envisagez l'intégration avec le système de paye. Décrivez les flux entre Chronos et le système de paye pour garantir la cohérence entre ceux-ci.

11) ***Génie logiciel (2 points)***

- Quels outils, méthodologie ou bonnes pratiques mettriez-vous en œuvre pour assurer une bonne couverture des différents cas ?
- Quelle infrastructure proposeriez-vous pour assurer une non-régression en cas de modifications ?

12) ***Évolutions (1,5 point)***

La DAS demande d'ajouter les réceptions du cadre des surveillants du Jardin du Luxembourg au système. Les surveillants du Jardin ne font jamais plus d'une réception par jour. Leurs feuilles de réception sont distinctes de celles des surveillants du Palais. Le chef du Service de surveillance du Jardin doit faire signer par le directeur de la DAS les récapitulatifs des primes de son service, indépendamment de ceux des surveillants du Palais.

Comment envisagez-vous cette évolution ?

ÉPREUVES ORALES D'ADMISSION

1. Épreuve portant sur des connaissances techniques

Cette épreuve est constituée par :

- un exposé oral d'une durée de *dix minutes* sur un sujet tiré au sort ;
- des questions, pendant *trente minutes*, ayant pour point de départ l'exposé oral et pouvant porter sur d'autres sujets.

(préparation 20 minutes – durée 40 minutes – coefficient 3)

2. Entretien libre avec le jury

Cette épreuve est constituée par :

- un exposé oral d'une durée de *cinq minutes* sur un cas concret tiré de l'expérience professionnelle du candidat (projet, stage ou travail d'étude) ;
- un entretien d'une durée de *vingt-cinq minutes* visant à apprécier l'adéquation des candidats à l'emploi d'informaticien et leur motivation pour exercer ces fonctions, ainsi que leur culture générale et leur perception des orientations et des enjeux des technologies de l'information.

Pour cette épreuve, le jury dispose d'une fiche de renseignements individuelle, préalablement remplie par les candidats et ne faisant l'objet d'aucune notation.

(durée 30 minutes – coefficient 5)

ÉPREUVE PORTANT SUR DES CONNAISSANCES TECHNIQUES

(Préparation 20 minutes – durée 40 minutes – coefficient 3)

Sujet n° 1 : Covoiturages de fin de séance

Lorsque les débats en séance publique durent au-delà de 21 heures, le Sénat prend en charge les frais de taxis, nécessaires pour ramener chez eux la centaine de personnels mobilisés qui le souhaitent. Pour limiter ces frais, le Bureau des transports (BT) de la direction de la Logistique et des Moyens généraux souhaiterait développer le partage des taxis entre personnels. Le principe consiste à regrouper jusqu'à trois personnes dont les destinations sont approximativement sur une même route, tout en évitant d'allonger excessivement le trajet de chacun.

Aujourd'hui, le BT prévoit le nombre de taxis nécessaires grâce à des fax que les personnels envoient en cours de soirée et qui indiquent si ces derniers acceptent de partager leur taxi. La quasi-totalité d'entre eux est d'accord sur le principe mais ce covoiturage est compliqué à mettre en place ; les personnels arrivent au point d'arrêt des taxis en ordre dispersé suivant leurs responsabilités. Il en est de même pour les taxis qui se présentent en fonction de leur localisation lors de la réservation. En général, les départs sont échelonnés dans l'heure qui suit la levée de la séance publique. Ainsi, les tentatives d'appariement se font de manière *ad hoc* avec un agent qui communique verbalement à toute la file d'attente les destinations des premiers. Mais la communication de la commune ou de l'arrondissement parisien de destination n'est pas suffisante pour déterminer précisément la pertinence de l'appariement. Par crainte d'un allongement du temps de trajet, la plupart des gens ne se manifestent finalement pas, même si cela pourrait leur permettre d'être prioritaire dans la file d'attente.

La facturation des courses est réalisée par la société de taxis prestataire directement auprès du Sénat, en fonction des trajets réalisés. Des copies des reçus signés sont délivrées aux personnes transportées.

Veuillez présenter une première analyse et une démarche à suivre. Quels sont les points importants ou critiques de ce projet ?

Sujet n° 2 : Dématérialisation

La commission des Affaires étrangères du Sénat suit les questions de politique étrangère et de défense. Elle examine notamment les projets de loi de ratification des traités et accords internationaux. Cette commission est composée de 57 sénateurs, assistés de 14 fonctionnaires.

La commission souhaite dématérialiser ses échanges avec le ministère des Affaires étrangères. Certaines informations, comme les notes de situation établies par le ministère sur un pays donné, ne doivent être accessibles qu'aux membres de la commission, notamment pendant les réunions en salle de commission. La salle de commission dispose d'un accès Wi-Fi qui permet l'accès à l'intranet du Sénat et aussi à Internet, mais aucun ordinateur n'y est installé. Un annuaire LDAP disponible dans la zone intranet du Sénat comprend tous les fonctionnaires du Sénat ainsi que tous les sénateurs. La communication avec le ministère ne devrait être possible que pour le sous-groupe des fonctionnaires rattachés à la commission et pour les sénateurs membres de la commission grâce à un login et un mot de passe personnels fournis par le ministère. On souhaiterait cependant qu'après l'initialisation du système, les personnes habilitées ne s'identifient qu'une seule fois, en utilisant leur login et leur mot de passe LDAP du Sénat.

Veuillez présenter une première analyse et une démarche à suivre. Quels sont les points importants ou critiques de ce projet ?

Sujet n° 3 : Gestion des concours

La direction des Ressources humaines et de la Formation (DRHF) est chargée d'organiser les concours d'entrée au Sénat.

Un concours est composé de phases (présélection, admissibilité, admission), elles-mêmes composées d'épreuves. Les épreuves au sein d'une phase sont affectées d'un coefficient et parfois d'une note éliminatoire.

Pour couvrir l'ensemble des métiers du Sénat, il existe une dizaine de concours dont les modalités diffèrent : conditions pour concourir, nombre de phases, nature des épreuves.

Il existe également des concours internes.

Un portail Web hébergé à l'extérieur permet aux candidats de se préinscrire depuis Internet en remplissant un formulaire. À la clôture des préinscriptions, la liste des préinscrits et les données du formulaire sont récupérées dans un tableau Excel.

Ce fichier Excel permet alors aux secrétaires de la DRHF d'effectuer la validation des inscriptions (contrôles de cohérence, vérification des pièces justificatives, détection des doublons).

Les inscriptions validées alimentent alors un autre fichier Excel, que l'administrateur-adjoint de la DRHF en charge du concours utilise pour saisir les notes des candidats aux épreuves, afin de procéder aux calculs des résultats et produire les classements.

La gestion des concours est un sujet sensible et ces fichiers Excel n'offrent pas un niveau de sécurité suffisant. D'une part, une perte de données est possible, ainsi qu'une altération soit des données, soit des formules de calcul. D'autre part, les candidats internes pourraient avoir accès aux données du concours. C'est pourquoi la DRHF souhaite abandonner ces fichiers Excel au profit d'une application spécifique de gestion des concours, développée par la DSI du Sénat. Cette application couvrira les processus suivants :

- la gestion des candidats ;
- le paramétrage des concours ;
- la saisie des notes des candidats ;
- le calcul des résultats par phase et du résultat final, en tenant compte des coefficients et des notes éliminatoires ;
- le classement par phase et final ;
- la production de multiples courriers (convocations, résultats) ;
- la confection de statistiques en tous genres.

Veuillez présenter une première analyse et une démarche à suivre. Quels sont les points importants ou critiques de ce projet ?

Sujet n° 4 : Projet vidéo

Le Sénat souhaite étoffer son offre de vidéo en constituant un portail vidéo rénové. Il désire diffuser tous ses travaux en direct et sous forme de vidéo à la demande (VOD), disponible après un certain délai. Ces vidéos doivent pouvoir être consultées sur un grand nombre de types de plateformes différents (smartphones, tablettes, ordinateurs portables, etc.).

La direction de la Communication a en charge la captation numérique et la diffusion des vidéos.

La sélection d'une vidéo, *via* une navigation arborescente ou *via* un moteur de recherche, de même que la navigation à l'intérieur d'une vidéo (par exemple pour arriver directement sur un point précis d'une séquence vidéo) reposent sur des métadonnées. Certaines sources de métadonnées peuvent potentiellement être exploitées, comme :

- la base Oracle des sénateurs et de leurs différentes appartenances (groupes politiques, commissions, etc.), tenue à jour par la direction de la Séance ;
- la base Oracle des textes législatifs en cours de discussion, tenue à jour par la direction de la Séance ;
- la base des travaux divers en cours, tenue à jour par la direction des Commissions ;
- l'agenda du Sénat, tenu à jour par différentes directions.

Par ailleurs, des saisies supplémentaires de métadonnées devront être réalisées.

Enfin, pour certaines vidéos qui concernent la séance publique, un suivi en temps réel des points de l'ordre du jour est disponible. La précision de ce suivi est inconnue.

Veuillez présenter une première analyse et une démarche à suivre. Quels sont les points (techniques et/ou organisationnels, etc.) importants ou critiques de ce projet ?