ANNEXE 1 : Tarif des terrasses année 2005

Mairie de P. - Service Emplacements - Cellule 118 - Mél : emplacements@ville-p.fr

OCCUPATION COMMERCIALE DU DOMAINE PUBLIC

Terrasses 2005

TARIFS 2005

> Terrasse permanente (du 1er janvier au 31 décembre - 365 jours)

Zone	Tarif (en € / m²)
A	126,50
B	106,36
C	74,74

> Terrasse semi-permanente (du 1er avril au 31 octobre - 214 jours)

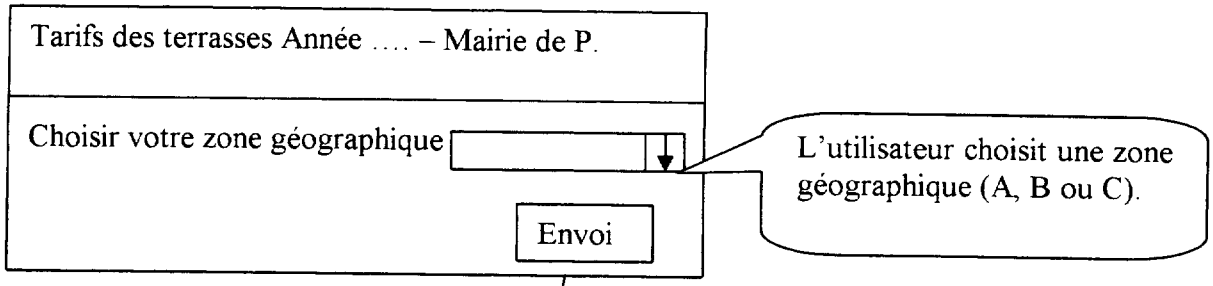
Zone	Tarif (en € / m²)
A	74,16
B	62,36
C	43,82

> Terrasse d'été (du 15 mai au 15 septembre - 123 jours)

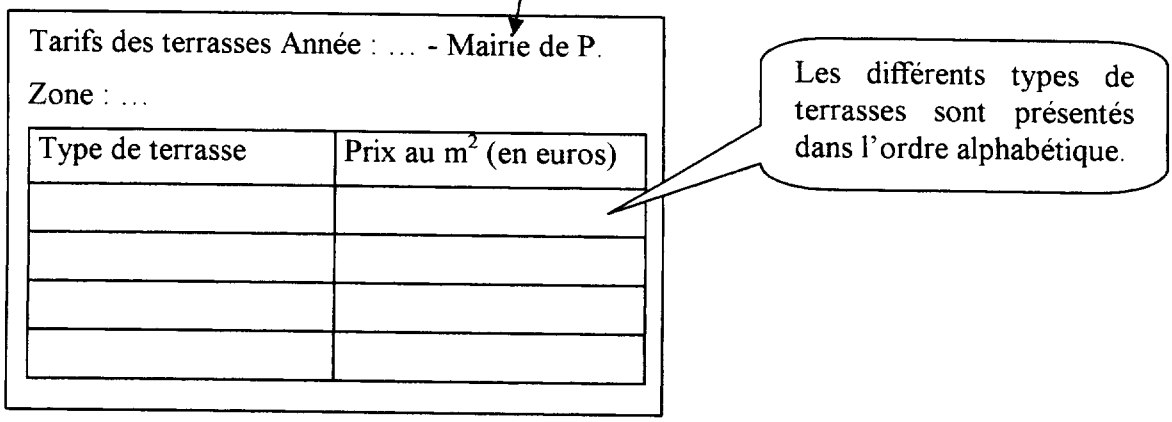
Zone	Tarif (en € / m²)
A	42,97
B	36,14
C	25,39

ANNEXE 2 : Maquette de la consultation des tarifs des terrasses sur internet

Page de sélection de la zone géographique



Page d'affichage des tarifs pour la zone géographique choisie



ANNEXE 3 : Aide HTML

Le code HTML pour afficher un tableau est fourni dans l'exemple ci-dessous :

```
<TABLE WIDTH="90%" BORDER>
<TR ALIGN="CENTER"><TH>Col1</TH><TH>Col2</TH><TH>Col3</TH></TR>
<TR ALIGN="CENTER"><TD>AA</TD><TD>BB</TD><TD>CC</TD></TR>
<TR ALIGN="CENTER"><TD>EE</TD><TD>FF</TD><TD>GG</TD></TR>
</TABLE>
```

Le résultat obtenu après interprétation de ces ordres est donné ci-dessous :

Col1	Col2	Col3
AA	BB	CC
EE	FF	GG

ANNEXE 4 : Ébauche de page d'affichage des tarifs

La page qui permet l'affichage des tarifs, en fonction de la zone choisie par l'internaute est construite dynamiquement, à partir d'instructions intégrées dans le code HTML, et interprétées par le moteur de script qui génère le code HTML correspondant. Les instructions sont écrites entre les balises <? et ?>.

```
<HTML>
    <BODY>
    Tarifs des terrasses Année :
    <? Afficher (Année(DateSystème())) ?>
    - Mairie de P.
    Zone : < ? Afficher (Zsaisie) ?>
    <BR>
    ...
    ...
    ...
    </BODY>
</HTML>
```

Script à compléter

La fonction *DateSystème()* renvoie une valeur de type *date* qui contient la date système.

La fonction *Année(date d)* est une fonction qui renvoie l'année de la date *d* passée en paramètre.

La variable *Zsaisie* est une variable caractère qui contient la valeur de la zone choisie (elle a été initialisée dans la page de sélection de la zone).

La fonction *Afficher* permet également de produire du code HTML à partir de variables et de texte. On utilise alors l'opérateur de concaténation +.

Ex : `<? Afficher (Zsaisie + "<BR>") ?>`

La fonction *Afficher* génère le code HTML comprenant la valeur de la variable *Zsaisie* suivie de la balise
.

Consignes pour l'écriture de la page

- Il n'est pas nécessaire de déclarer les variables.
- Pour récupérer les données à afficher dans le tableau présentant les tarifs par type de terrasse, on utilise une procédure déjà définie dont l'en-tête est le suivant :

```
Procédure RechTarifs (     entrée zone : Caractère,
                           sortie tabRes : tableau de 1 à 10 de Ttarif,
                           sortie nbEl : entier)
```

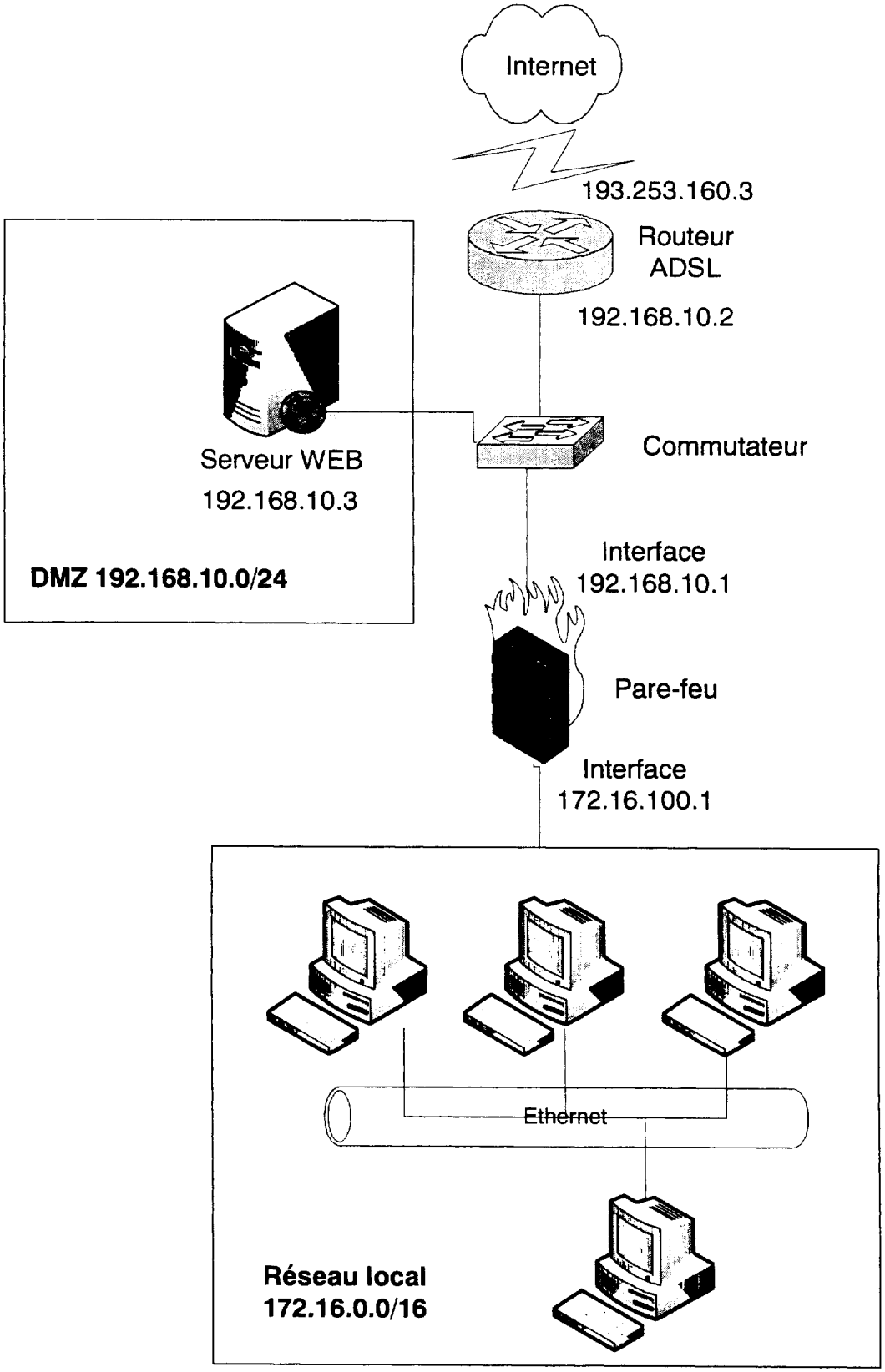
Cette procédure va rechercher dans la base de données les types de terrasse et leur tarif en fonction de la zone passée en paramètre. Elle admet comme paramètres en entrée la zone, en sortie le tableau contenant pour chaque élément un type de terrasse et un prix, ainsi que le nombre d'éléments retournés dans le tableau. Le tableau est trié par ordre alphabétique de type de terrasse. On estime qu'il existe au moins un type de terrasse et qu'il n'existe pas plus de dix types de terrasses dans la base de données.

Chaque élément du tableau a un type structuré déclaré comme suit :

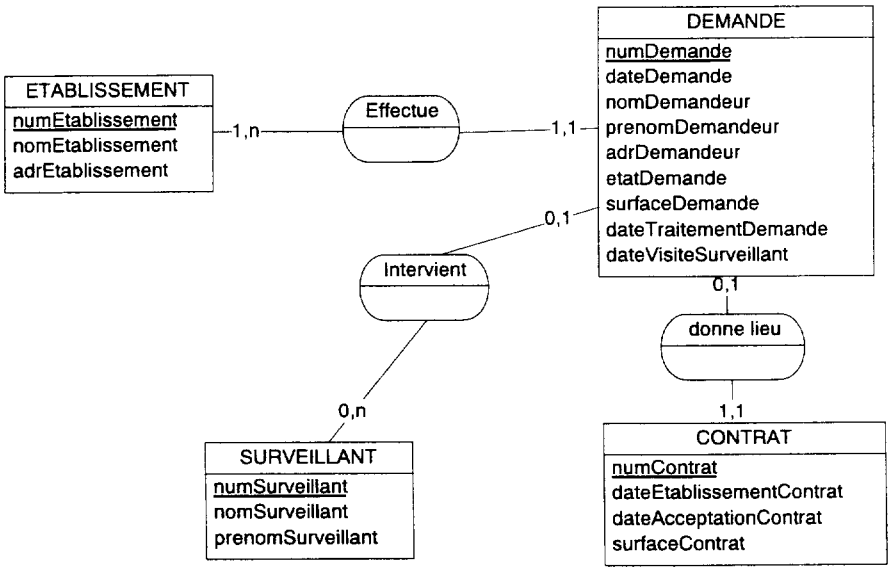
```
Ttarif : type structuré
     typeTer : chaîne de caractères
     prix : Réel
fin type structuré
```

La définition de la procédure ainsi que la déclaration du type structuré sont écrites dans un fichier inclus au début de la page. *Cette inclusion n'est pas demandée.*

ANNEXE 5 : Réseau informatique de la Mairie



ANNEXE 6 : Schéma entité-association pour la gestion des demandes de terrasse



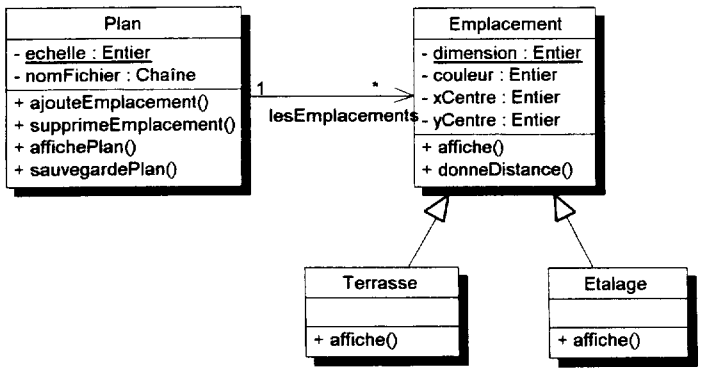
Remarque : Dans l'entité DEMANDE, la propriété *etatDemande* permet de connaître l'état de traitement de la demande d'installation de terrasse. Cette propriété peut prendre l'une des valeurs suivantes :

- "prête" (état initial)
- "non recevable",
- "mise en attente",
- "en cours"
- "refusée",
- "satisfaite",
- "sans suite".

ANNEXE 7 : Représentation des classes

Diagramme de classes :

Afin de ne pas alourdir le schéma, seuls figurent les membres (données et méthodes) significatifs. Les paramètres des méthodes ne sont pas présentés.



Commentaires :

Accessibilité :

Symbole – (moins) : membre privé

Symbole + (plus) : membre public

Un membre souligné est un membre à portée classe, appelé également membre de classe ou *static*.

Héritage :

Les classes *Terrasse* et *Etalage* héritent de la classe *Emplacement.*

Association entre Plan et Emplacement :

Représente l'ensemble des emplacements rattachés au plan.

Déclaration des classes correspondant au diagramme de classes

Dans les fonctions ou procédures, les paramètres sont précédés de **e** *pour « entrée »,*

```
classe Plan
' Attribut privé à portée classe
      echelle : Entier
' Attributs privés
      nomFichier : Chaîne
      lesEmplacements : Collection de Emplacement
        ' Mémorise tous les emplacements placés sur le plan (terrasse ou étalage)
' Méthodes publiques
      fonction ajouteEmplacement(e unEmplacement : Emplacement) :
Booléen
        ' Ajoute le nouvel emplacement (terrasse ou étalage) dans la collection lesEmplacements. Elle renvoie une valeur
        booléenne permettant de savoir si l'ajout a été possible.
      procédure supprimeEmplacement(e unEmplacement : Emplacement)
        ' supprime l'emplacement de la collection lesEmplacements. Elle vérifie au préalable l'existence de l'emplacement dans
        la collection.
      procédure affichePlan()
        ' Affiche l'image bitmap du plan avec les emplacements (terrasse et étalage). On distingue une terrasse par un rond vert
        et un étalage par un carré rouge.
      procédure sauvegardePlan()
        ' Sauvegarde le plan et ses emplacements dans le fichier
Fin classe Plan
```

classe Emplacement

// Cette classe hérite d'une méthode `estType(<nomClasse>) : Booléen` de la super-classe `Objet` permettant de savoir si l'objet invoquant cette méthode appartient à la classe dont le nom est passé en paramètre. Exemple d'utilisation : L'appel `monObjet.estType(maClasse)` renvoie `vrai` si `monObjet` est une instance de `maClasse`, `faux` sinon.

// Attribut privé à portée classe

`dimension : Entier` *// rayon pour le cercle, largeur pour le carré*

// Attributs privés

`couleur : Entier`

`xCentre : Entier` *// abscisse du centre de l'emplacement sur le plan, cercle ou carré*

`yCentre : Entier` *// ordonnée du centre de l'emplacement sur le plan, cercle ou carré*

// Méthodes publiques

procédure `affiche()`

fonction `donneDistance(e unEmplacement : Emplacement) : Entier`

// Renvoie la distance en mètres séparant l'emplacement courant d'un autre emplacement passé en paramètre.

fin classe Emplacement

classe Terrasse hérite de Emplacement

// Méthode publique

procédure `affiche()` *// Affiche un cercle*

fin classe Terrasse

classe Etalage hérite de Emplacement

// Méthode publique

procédure `affiche()` *// Affiche un carré*

fin classe Etalage

L'implémentation de la classe **Plan** utilise une classe technique **Collection** qui offre des services pour la gestion d'un ensemble d'objets.

classe Collection // Méthodes publiques

Fonction `cardinal() : Entier` *// Renvoie le nombre d' éléments de la collection*

Fonction `existe(e unObjet : Objet) : Booléen`

// Teste si unObjet existe dans la collection

Fonction `index(e unObjet : Objet) : Entier`

// Renvoie l'index de unObjet, le premier objet de la collection a pour index 1

Fonction `extraireObjet(e unIndex : Entier) : Objet`

// Retourne l'objet d'index unIndex

Procédure `ajouter(e unObjet : Objet)` *// Ajoute un objet à la collection*

Procédure `enlever(e unIndex : Entier)` *// Supprime l'objet d'index unIndex de la collection*

Procédure `vider()` *// Vide le contenu de la collection*

Fin classe Collection

ANNEXE 8 : Informations pour le calcul de la charge de l'étape de programmation

Nombre de jours-hommes (jh) nécessaires pour l'élaboration d'un type d'écran ou d'état imprimé en fonction du niveau de complexité

	Simple	Moyenne	Importante
Écran de consultation	3 jh	4 jh	6 jh
Écran de mise à jour	4 jh	5 jh	10 jh
État imprimé	1 jh	4 jh	12 jh

Remarques sur la qualification des écrans :

- La complexité « Simple » correspond à une fenêtre contenant moins de 16 composants ergonomiques contenant des données métiers.
- La complexité « Moyenne » correspond à une fenêtre contenant entre 16 et 25 composants ergonomiques contenant des données métiers.
- La complexité « Importante » correspond à une fenêtre contenant un classeur d'onglets.

En affectant une charge unitaire à chaque objet à livrer, on obtient une charge qui couvre les spécifications techniques, la programmation et les tests unitaires.

On doit alors y ajouter :

- 10 % de la charge obtenue pour les tests d'enchaînement (ou d'intégration) ;
- 20 % de la nouvelle charge obtenue pour l'encadrement de cette phase.

Extrait du devis de sous-traitance du développement de l'interface homme-machine

Durée de réalisation : 18 jours.
Coût total : 19 000 € HT.

Éléments d'évaluation des charges de personnel au sein du service informatique de la Mairie de P.

1 jour-homme, quel que soit le statut de la personne, est évalué à 200 €.