CartoSpip

TELECOM PARIS
école nationale supérieure des télécommunications

CartoSpip

Web cartographique Coopératif

Conception logicielle

Encadrement et supervision

Annie Danzart
Jean-Claude Moissinac

Equipe de développement

Mohamed Anouar Benaissa
anouar@benaissa.net

Anthony Rabiaza
rabiaza@enst.fr

Déva Pajaniaye
pajaniay@enst.fr

Mastères Spécialisés 2005
Ingénierie du logiciel / Conception et Architecture des systèmes informatique

PLAN

Architecture du logiciel

Organisation des composants

Nos fichiers Java seront organisés de la manière suivante :

Les différentes classes (Beans, Servlet ou librairies d'accès à la base) sont contenues dans le package **enst.cartospip**.

1. Les fichiers *sources* seront dans le répertoire /CartoSpip :

	Sources	*Emplacement*
Fichiers JSP	*.jsp	/
Beans	*Beans*_xxx.class	/src/enst/cartospip
Servlet	*.java	/src/enst/cartospip
Applet	*.java	/Applet/src

2. Les fichiers produits après *compilation* seront dans le répertoire Webapps de Jarkarta Tomcat :

	Executables	*Emplacement*
Fichiers JSP	*.jsp	/ROOT/
Beans	*Beans*_xxx.class	/ROOT/WEB-INF/ classes/enst/cartospip
Servlet	*.class	/ROOT/WEB-INF/ classes/enst/cartospip
Applet	*.class	/ROOT/

Pages JSP et Beans

Le site web CartoSpip comporte 5+1 pages (JSP) :

Interface Client

→ Affichage
→ Consultation
→ Modification
→ Authentification

→ Validation

→ Home page
Correspondant à la page d'authentification,

→ Page d'accueil
Confirmant l'authentification avec un message de bienvenue

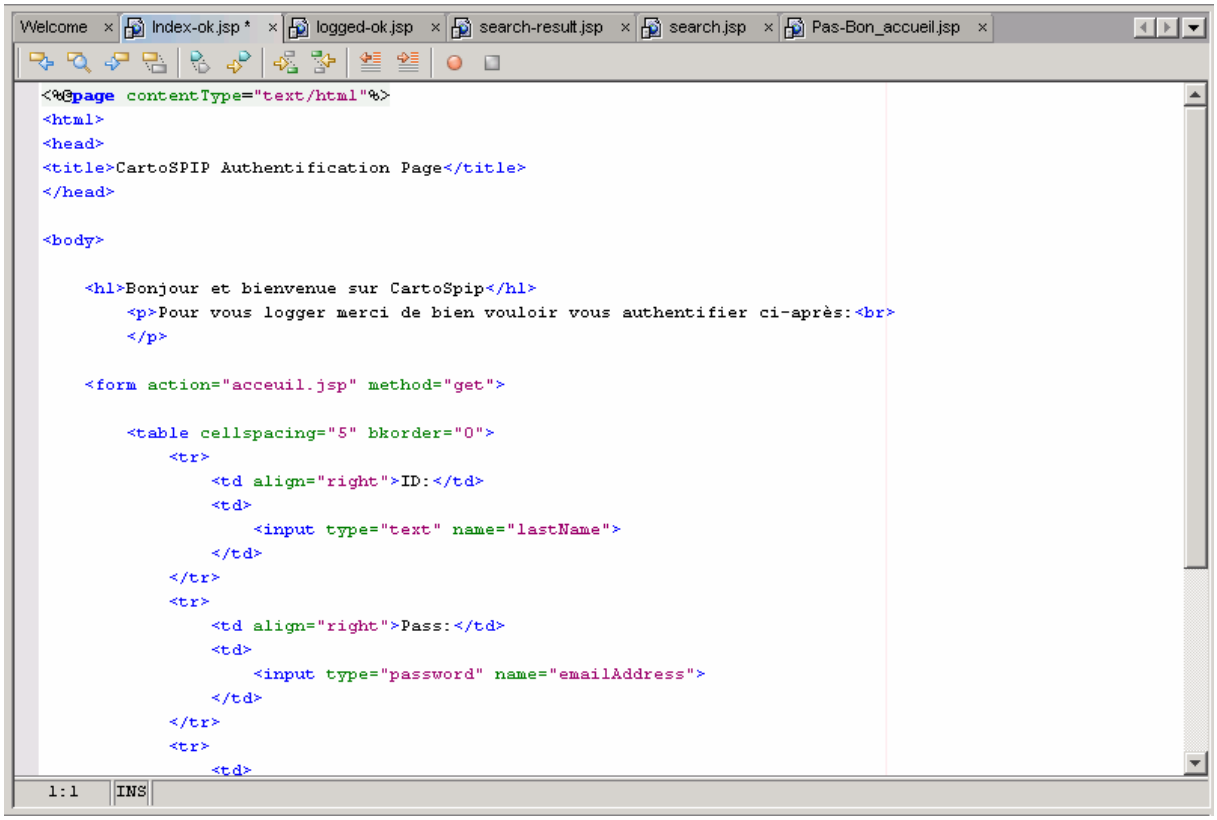
→ Page de recherche
Propose un champ de recherche qui permettra de demander une carte vide (sans annotation) d'un endroit donné.

→ Page de résultats
Donne accès évidemment aux résultats de la recherche

→ Page de consultation
Offre un filtrage par mot-clef.

→ Page d'aide
Fourni un descriptif détaillé des fonctionnalités offertes par la plateforme CartoSpip dont un résumé est visible sur la page d'accueil.

→ Une page de validation des cartes
Si l'utilisateur c'est authentifié en tant que SuperUser,

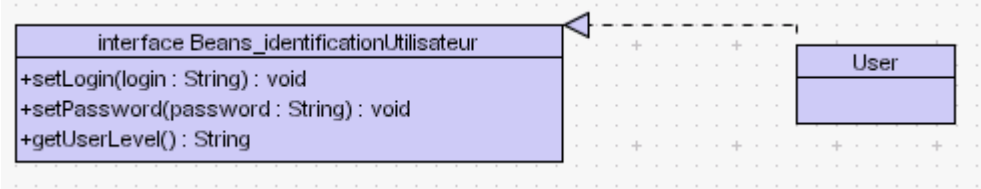


Beans

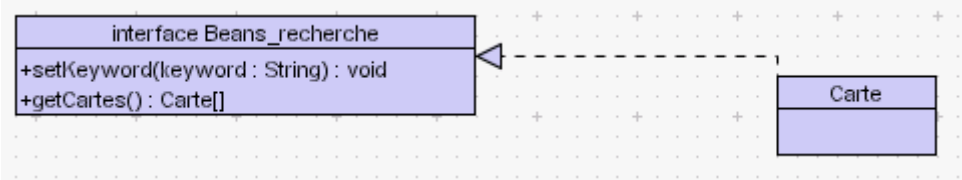
Les Beans sont les composants qui seront utilisés dans les pages JSP pour accéder aux différentes informations contenues dans la base de données :

Les JavaBeans obéissent à des conventions de nommage quant à ses méthodes d'accès aux attributs et à ses événements. Leurs manipulations seront alors aisées et transparentes au niveau du JSP.

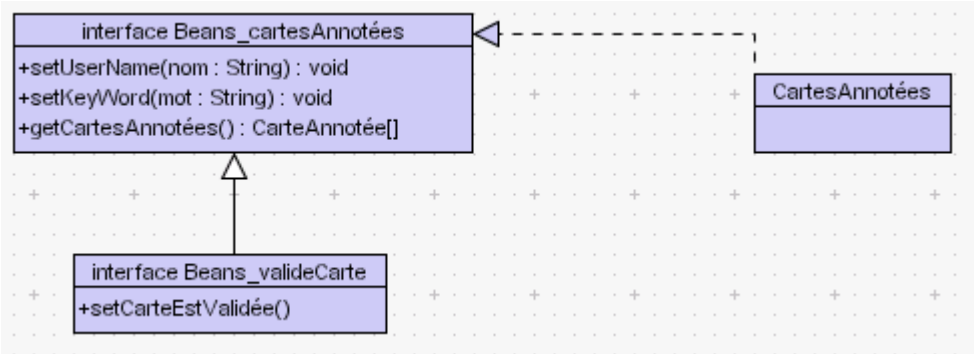
A. Beans identification (au niveau de la page d'accueil)



B. Beans recherche (au niveau de la page résultats de recherche)

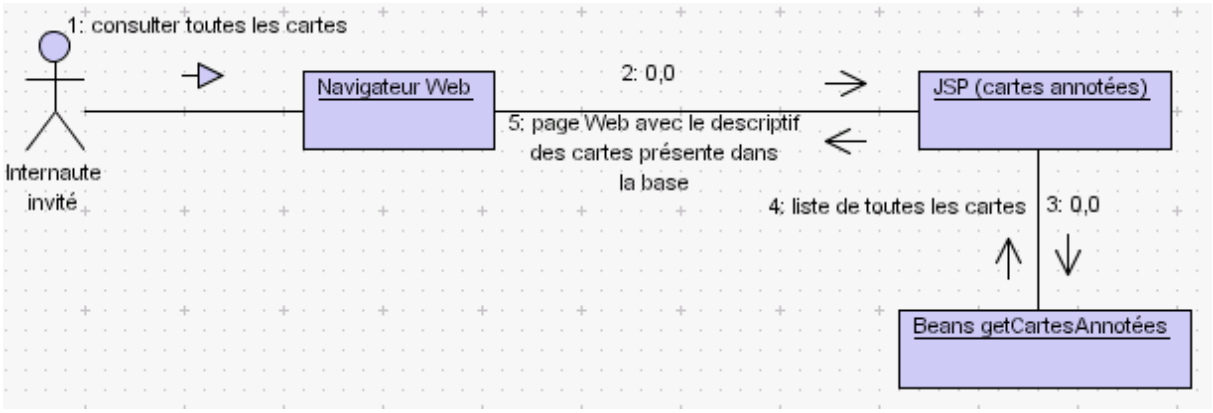


C. Beans pour l'affichage des cartes annotées par un utilisateur enregistré
Beans pour l'affichage de l'ensemble des cartes annotées visualisables
Beans de validation des cartes pour un « super user »



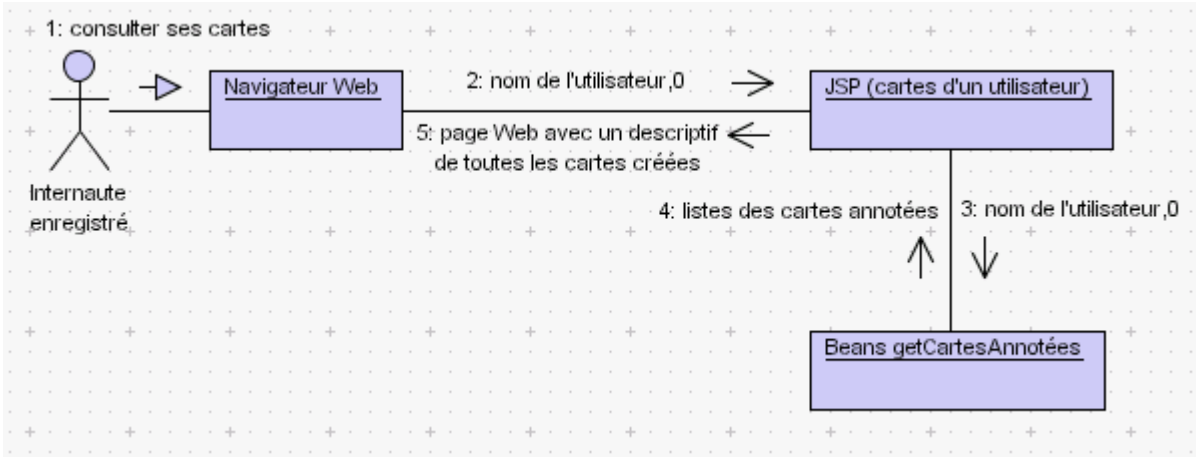
Intégration des Beans

Voici la manière dont s'effectue l'intégration des Beans dans les pages JSP dans le cas d'un listage des cartes annotées :



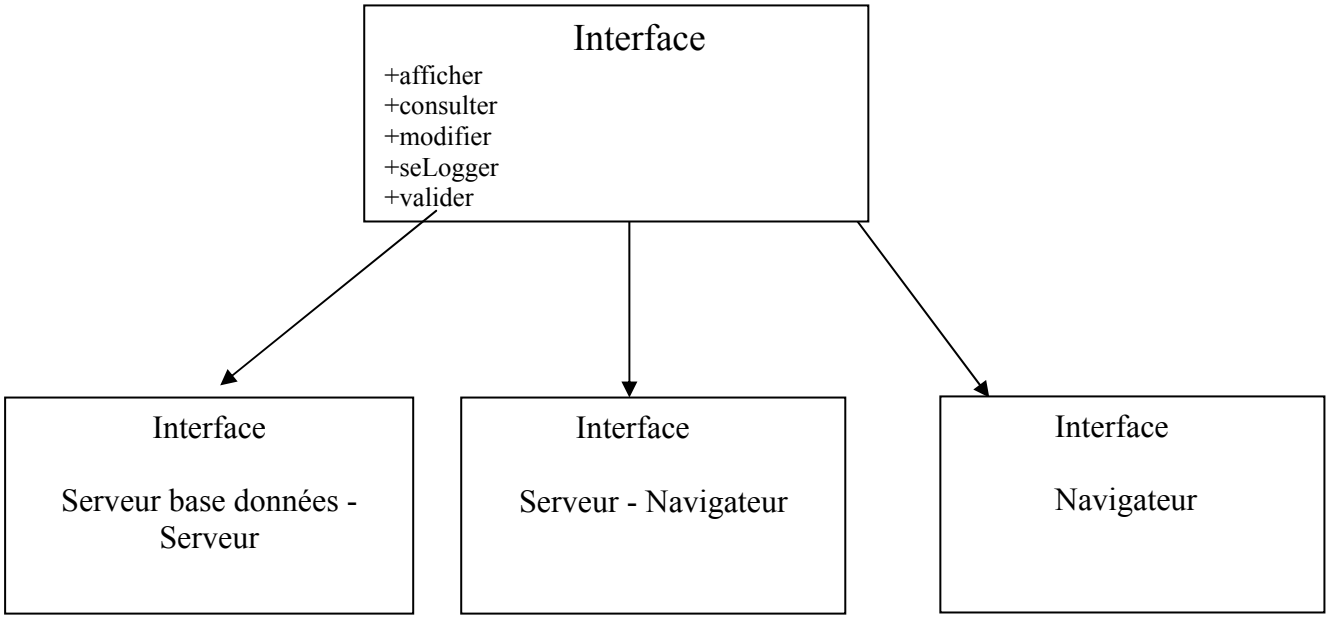
Remarque : Les messages 2 :0,0 et 3 :0,0 ne fournissent ni le nom de l'utilisateur ni le mot clé filtrant pour l'extraction des cartes annotées au Beans, celui-ci listera alors l'intégralité de la base de cartes annotées.

De la même manière, lorsque qu'un utilisateur voudra consulter ses cartes, nous avons le diagramme de collaboration suivant :

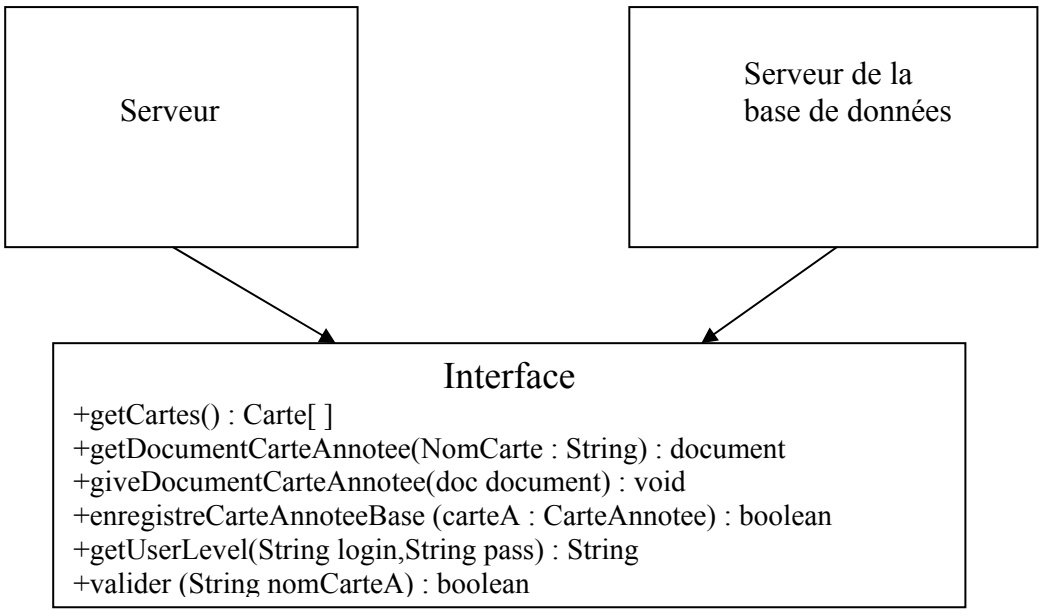


Remarque : Ici nous fournissons le nom de l'utilisateur. Le Beans ne va alors lister que les cartes pour cet utilisateur donné.

Déploiement de la base de données



Serveur et base de données



Les déclarations de classes :

```
public Class Carte {
  private String Nom;
  private String Url;
  private String Commentaires;
}

public Class CarteAnnotee {
    private Carte LaCarte;
    private String NomCarte;
    private String Auteur;
    private point RefPoint1;
    private point RefPoint2;
    private Objet_Graphique[] Objets_graphiques;
    private String Commentaires;

}

public class Objet_Graphique {
  private String Label ;
  private String Couleur;
  private String Url;
  private String Type;
  private String Commentaires;
  private point Coord;

}

public class Utilisateur{
  private String login;
  private String pass;
  private String Nom;
  private String Prenom;
  private String Titre;
  private int Tel;
  private String Adresse ;

}


Public Valideur extends Utilisateur{
Public validerCarteAnnotee(String NomCarteAnnotee) {

  }
}
```

Les méthodes essentielles

Cette méthode fournit toutes les cartes vierges de la base de données sous forme d'un tableau de carte
/* retourne Carte[] : tableau de Carte
public Carte[] getCartes(){

}

Cette méthode renvoie le type d'un utilisateur grâce à son login et son mot de passe passé en paramètre
public String get_User_Level (String login, String pass) {
/* connexion à la base */
/* sélection du type dans la base */
Return TypeUser ;
}

Fournit les informations d'une carte annotée sous forme de document XML
retourne un document XML contenant une carte Annotée avec ses objets graphiques à partir du nom de carte passé en paramètre
public document getDocumentCarteAnnotee (String nomcarteA) {
document doc ;
…..
Return doc ;
}

/* méthode qui parse le document Xml d'une CartesAnnotee et met à jour les attributs de l'instance de carteAnnotee appelant en focntion de son contenu
Arg : document contenant la carte Annotee
parseDocumentXml (document doc){

}

Cette méthode vérifie si un élément existe dans une table
Argument : l'élément et le nom de la table
Utilité : éviter le nom respect des contraintes au niveau de la clé primaire quand on veut insérer un élément dans une table
public boolean existe (String element, String tablename) {
…
Return resut;
}

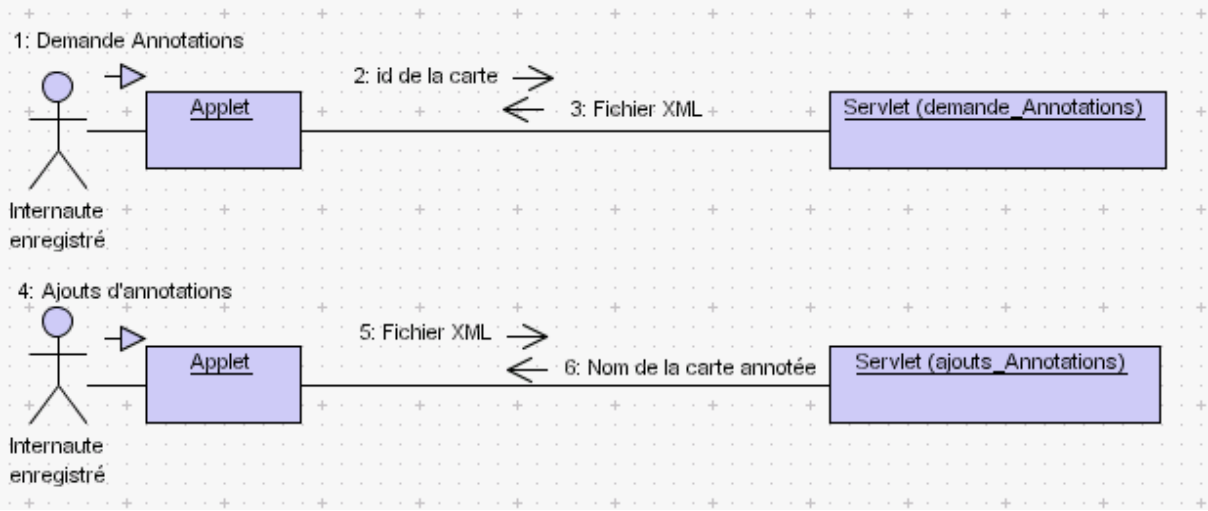
Méthode qui enregistre une carte Annotee passé en arg dans la base de données des cartes A
Elle a comme argument la carteA : CarteAnnotee
Retourne : True si réussite False sinon
public boolean enregistreCarteAnnoteeBase (CarteAnnotee nomCarteA) {
…
}

Méthode qui extrait une carte Annotee de la base de données des cartes A à patir de son nom
Elle a comme argument le nom de la carte Anootée NomCarteA : CarteAnnotee
Retourne : True si réussite False sinon
public boolean extraitCarteAnnoteeBase (String nomCarteA) {
…
}

Servlet

Les Servlets assurent la communication entre l'applet et le serveur. Les données sont transmises en HTTP/XML. Les Servlets jouent le rôle de pivot entre la base de données et l'internaute qui veut récupérer ou ajouter des annotations sur une carte.

Dans le cas d'une demande d'annotations, voici le diagramme de collaboration pour le cas de la visualisation d'une carte annotée et dans le cas de rajouts d'annotation :



Le *servlet* demande_Annotations :

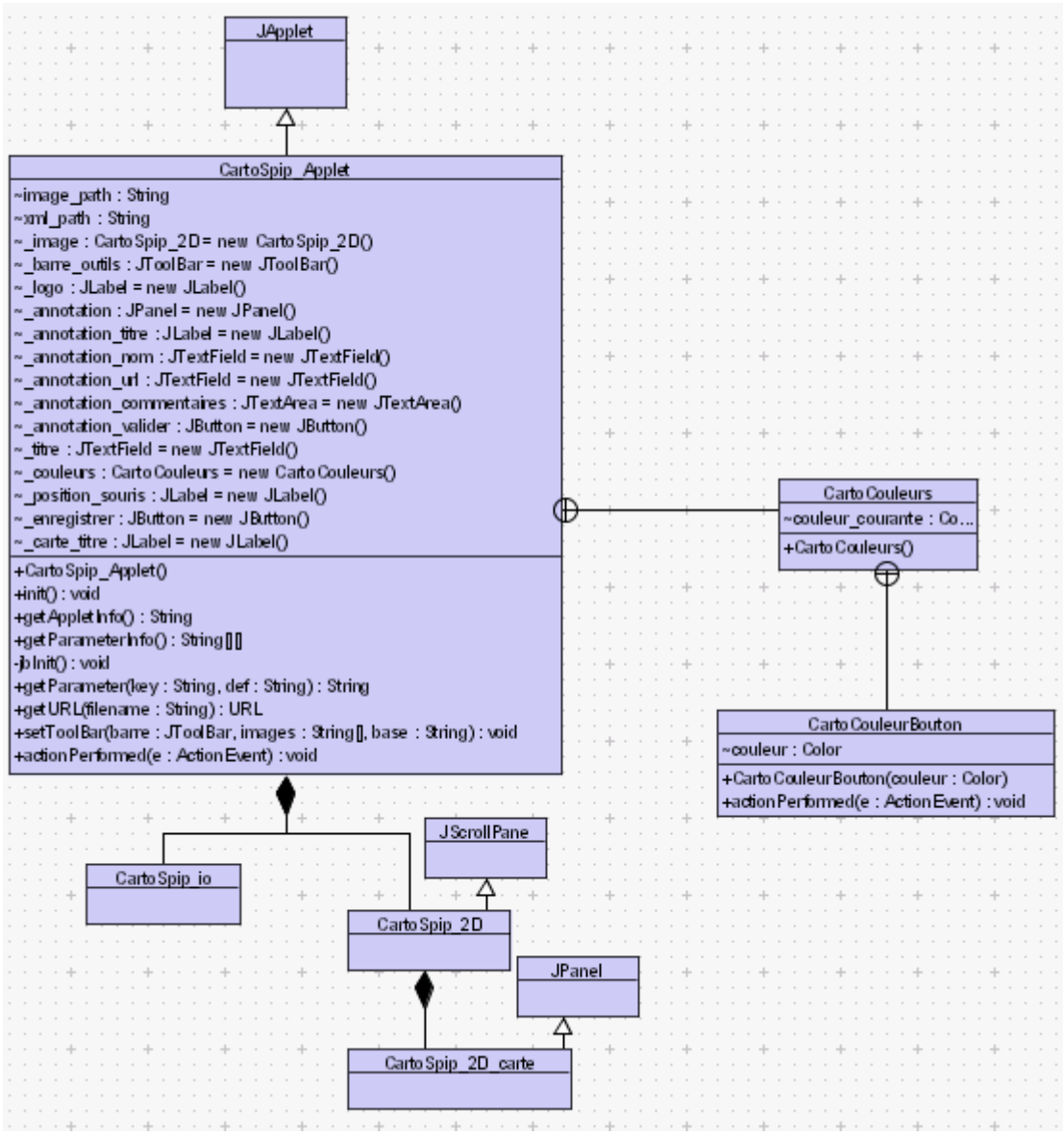
- Récupère les résultats des requêtes effectuées sur la base de données (pour une carte donnée) sous forme de document XML.
- Le fournit à l'applet sous forme de flux HTTP.

Le *servlet* ajouts_Annotations :

- Réceptionne un document XML envoyé par l'applet
- Fait appel aux librairies relatives à la base de données pour « parser » le fichier et effectuer les insertions de tuples correspondantes.
- Envoi à l'applet le nom de la carte annotée (ou un code d'erreur).

Applet

L'applet Java (CartoSpip_Applet) est composée d'un module principal contenant les différents widgets d'affichage textuels sur les url, les commentaires et celle sur la navigation : mode souris, mode déplacement, mode insertion de drapeaux ou de zones. Elle intègre une palette de couleurs (CartoCouleurs) composées de boutons de sélection de couleur (CartoCouleurBouton).
L'applet intègre un widget d'affichage de carte (CartoSpip_2D) et également un composant qui permet la gestion (lecture, création et envoi) de flux XML.



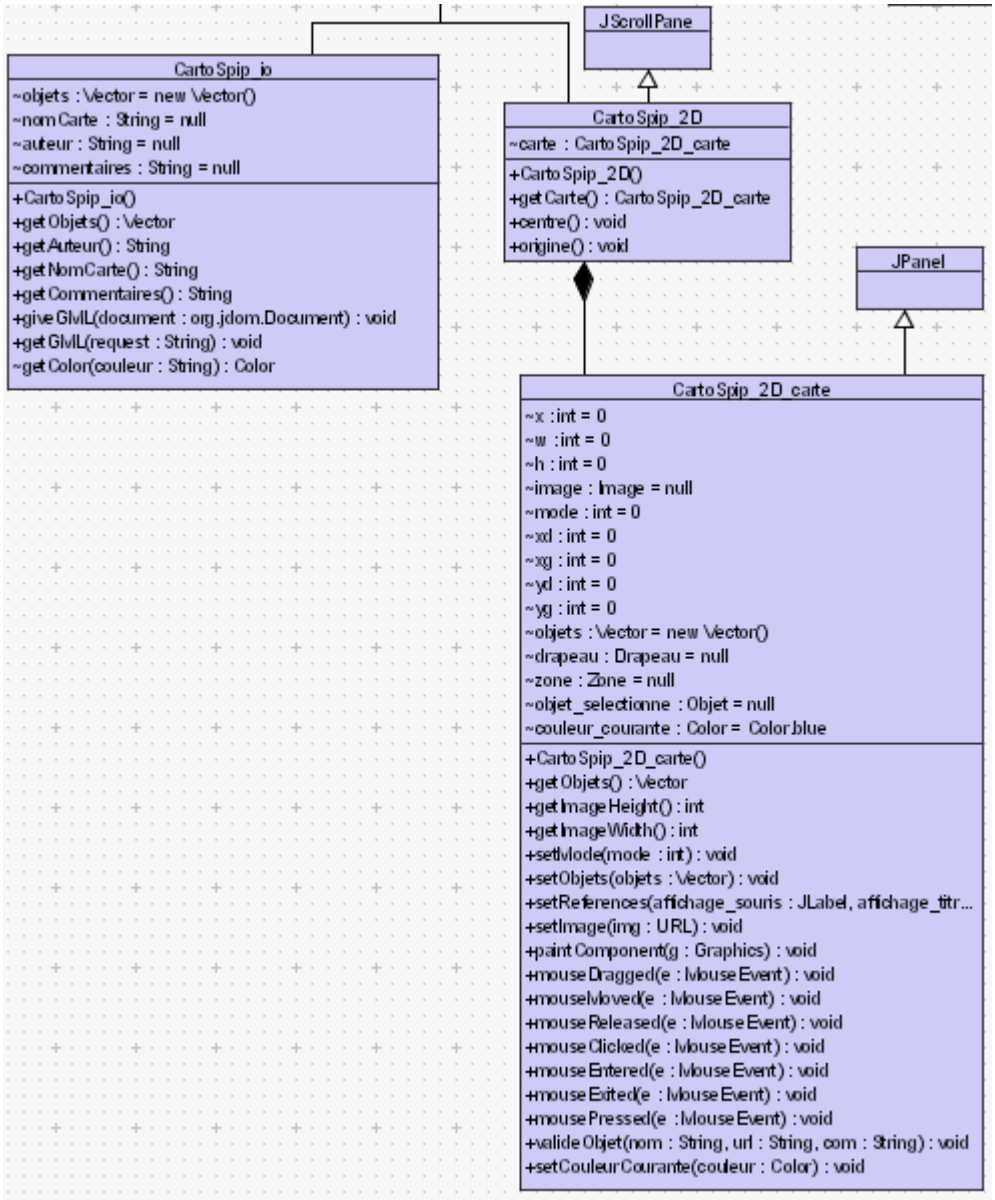
Composant de communication et widget d'affichage

CartoSpip_io (composant de communication)

Ce composant communique avec le serveur d'application. Il permet de récupérer un fichier distant (getGML()), de « parser » le fichier XML et d'extraire les informations sur la carte annotée (nom de l'auteur, commentaires) et sur objets graphiques (drapeaux et zones).

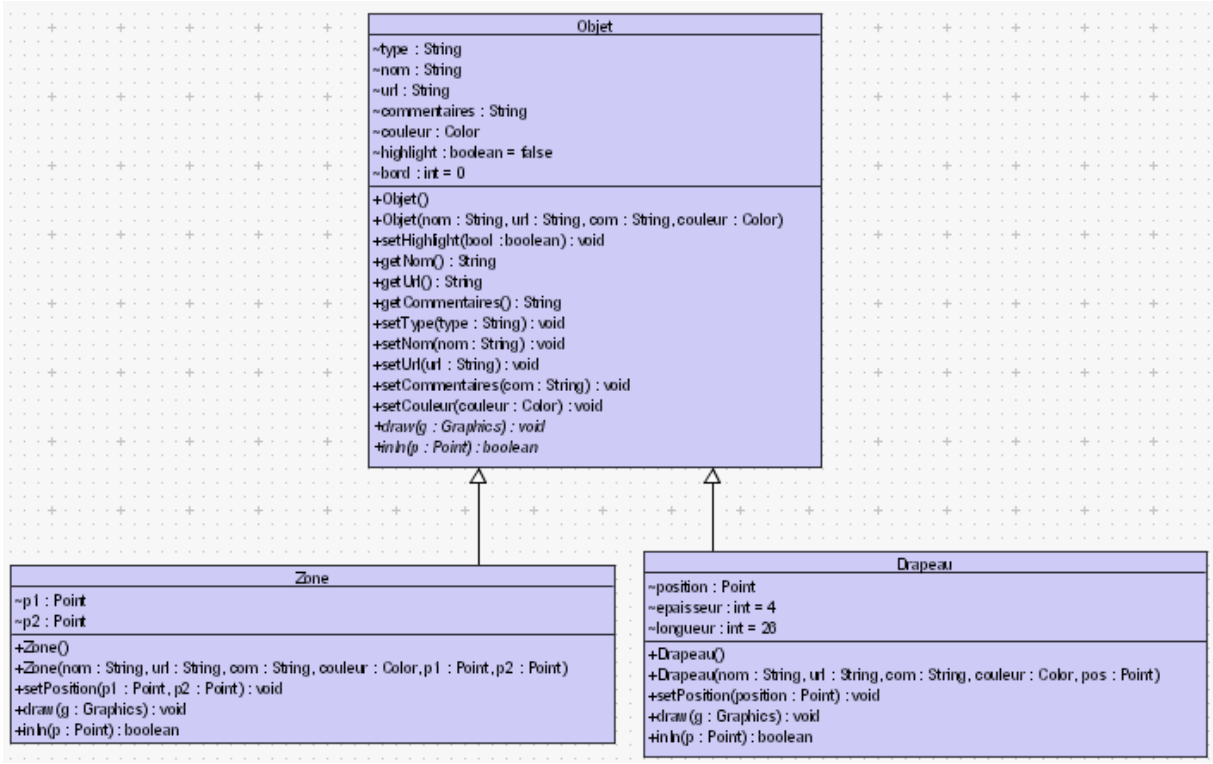
CartoSpip_2D_carte (widget d'affichage de carte et d'objets graphiques)

Ce widget s'occupe de l'affichage des objets graphiques (Vecteur *objets*). Ce widget étend le composant Swing javax.swing.JPanel et implémente MouseListener et MouseMotionListener pour la gestion des actions de la souris (insertions de nouveaux objets ou sélection d'un objet).

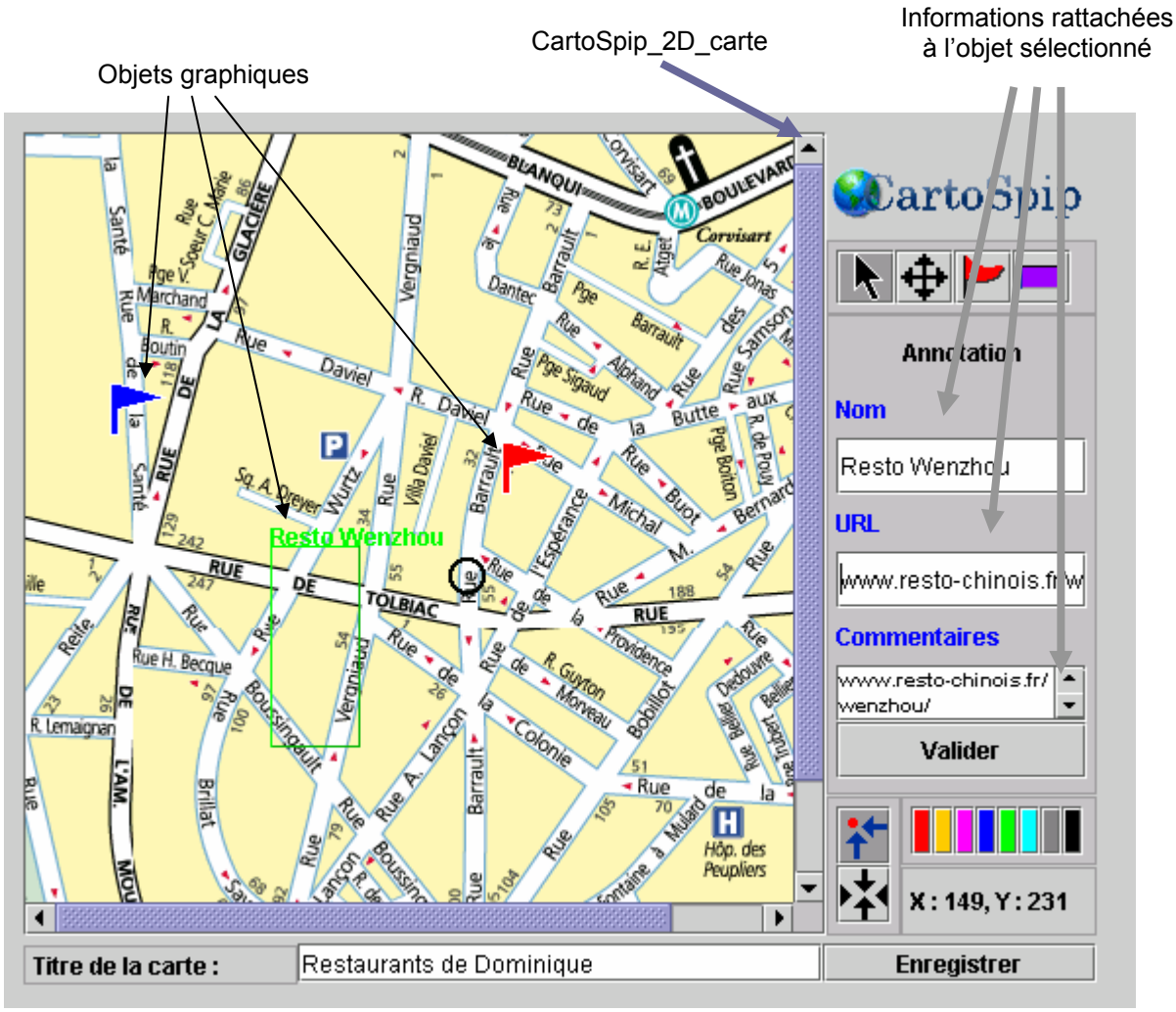


Objets graphiques : diagramme des classes

Voici le diagramme des classes représentants les objets graphiques :
Les objets graphiques Zone et Drapeau hérite de la classe *abstraite* Objet.



L'applet Java et les modules utilisés



Dialogue Applet/Serveur

Voici un exemple de fichier échangé entre ces deux entités, ce document XML respecte une DTD (cartospip.dtd). Il respecte également le standard de l'OpenGIS Consortium.

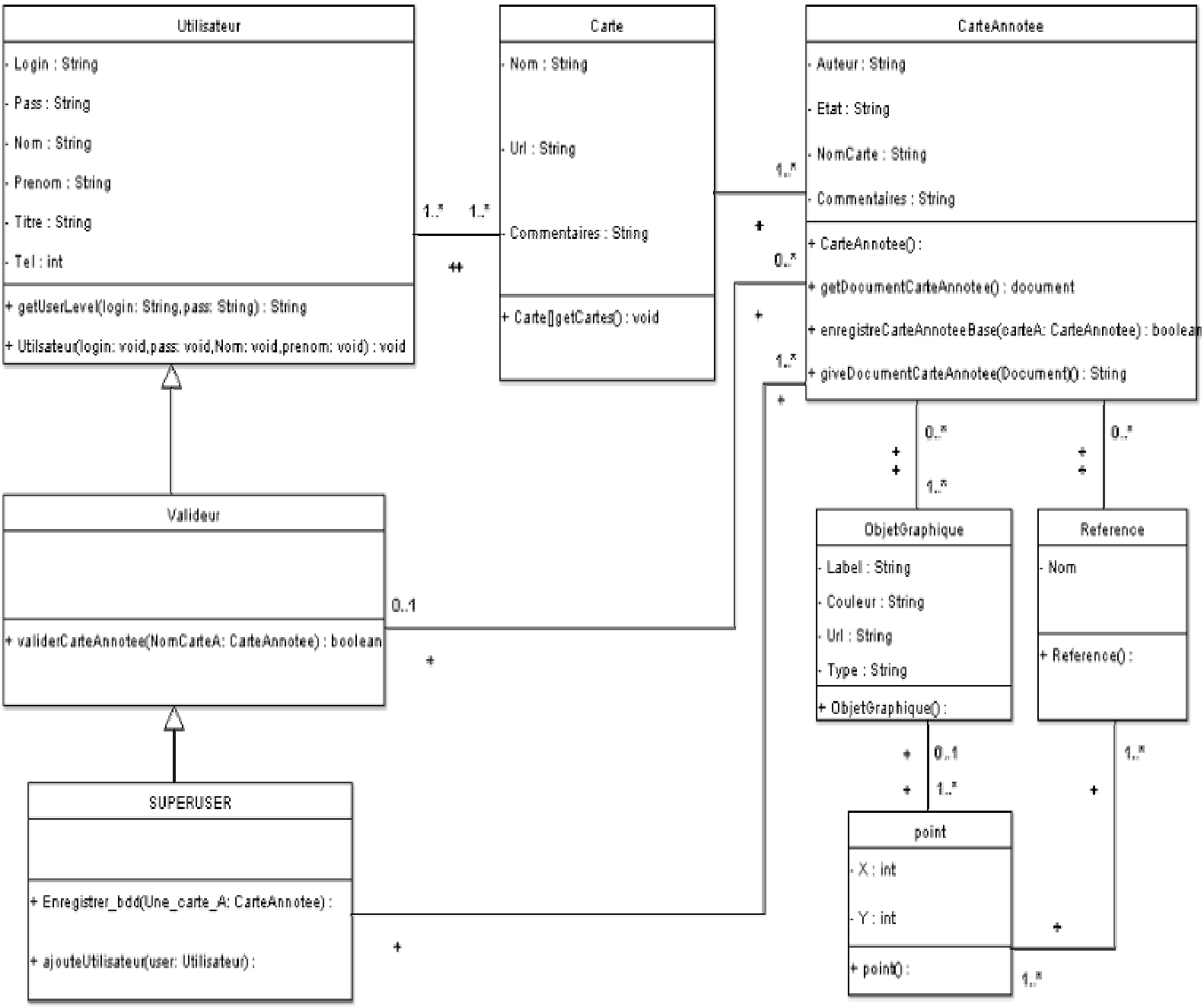
Deux namespaces seront utilisés :

- Un namespace relatif à cartospip (**cartospip :**)
- Celui de GML 2.0 (**gml :**).

```
<!DOCTYPE cartospip:carteAnnotee SYSTEM "cartospip.dtd">
<cartospip:carteAnnotee xmlns:cartospip="cartospip.dtd" xmlns:gml="http://www.opengis.net/gml">
        <cartospip:auteur>JP</cartospip:auteur>
        <cartospip:nomCarte>Restaurants de Dominique</cartospip:nomCarte>
        <cartospip:commentaires>Restos du 13eme arrondissement</cartospip:commentaires>
        <!-- Bounding-Box -->
        <cartospip:ref>
                <gml:Point srsName="" gml:id="">
                        <gml:coord>
                                <gml:X>1</gml:X><gml:Y>1</gml:Y>
                        </gml:coord>
                </gml:Point>
                <gml:Point srsName="" gml:id="">
                        <gml:coord>
                                <gml:X>30</gml:X><gml:Y>30</gml:Y>
                        </gml:coord>
                </gml:Point>
        </cartospip:ref>
        <!-- Objet(s) -->
        <!-- drapeau -->
        <cartospip:objet type="drapeau" label="Le wang">
                <!-- URL -->
                <cartospip:url>www.resto.fr/Le_Wang/</cartospip:url>
                <!-- Commentaires -->
                <cartospip:commentaires>
                        Le Wang : très bonne cuisine (je vous conseille les soupes Phô)
                        M°:Corvisart
                </cartospip:commentaires>
                <!-- bleu -->
                <cartospip:couleur>bleu</cartospip:couleur>
                <gml:Point srsName="" gml:id="">
                        <gml:coord>
                                <gml:X>5</gml:X>
                                <gml:Y>100</gml:Y>
                        </gml:coord>
                </gml:Point>
        </cartospip:objet>
        <!-- zone -->
        <cartospip:objet type="zone" label="Resto Wenzhou">
                <cartospip:url>www.resto-chinois.fr/wenzhou/</cartospip:url>
```

Architecture du serveur

Schéma UML



```
/*Fonction qui ajoute un objet graphique à un élément passé en paramètre*/
public void ajouterPointAElement(Element e, point UnPoint){

   Element p                        = new Element("Point");
   Attribute srsname1               = new Attribute("srsName","");
   p.setAttribute(srsname1);
   Attribute srsname2               = new Attribute("id","");
   p.setAttribute(srsname2);

   Element coord                    = new Element("coord");
   p.addContent(coord);

   Element x                        = new Element("X");
   Element y                        = new Element("Y");

   x.setText(""+UnPoint.getX());
   y.setText(""+UnPoint.getY());
   coord.addContent(x);
   coord.addContent(y);

   e.addContent(p);
}
```

```
/*fonction qui crée le document XML à partir des attributs de la classe carte Annotée*/
public Document getDocumentXml(){
     //Nous allons commencer notre arborescence en créant la racine XML qui sera ici "carteAnnotée"
     Element racine = new Element("carteAnnotee");

     //On crée un nouveau Document JDOM basé sur la racine que l'on vient de créer
     org.jdom.Document document     = new Document(racine);
     Element auteur                 = new Element("auteur");
     auteur.setText(Auteur);
     racine.addContent(auteur);

     Element nomCarte               = new Element("nomCarte");
     nomCarte.setText(NomCarte);
     racine.addContent(nomCarte);

     Element commentaires           = new Element("commentaires");
     commentaires.setText(LaCarte.getCommentaires());
     racine.addContent(commentaires);

     Element ref                    = new Element("ref");
     racine.addContent(ref);
     ajouterPointAElement(ref,RefPoint1);
     ajouterPointAElement(ref,RefPoint2);

     for (int i=0 ; i< Objets_graphiques.length ; i++)
       ajouterObjetAElement(racine, Objets_graphiques[i]);
   return document;
}
```

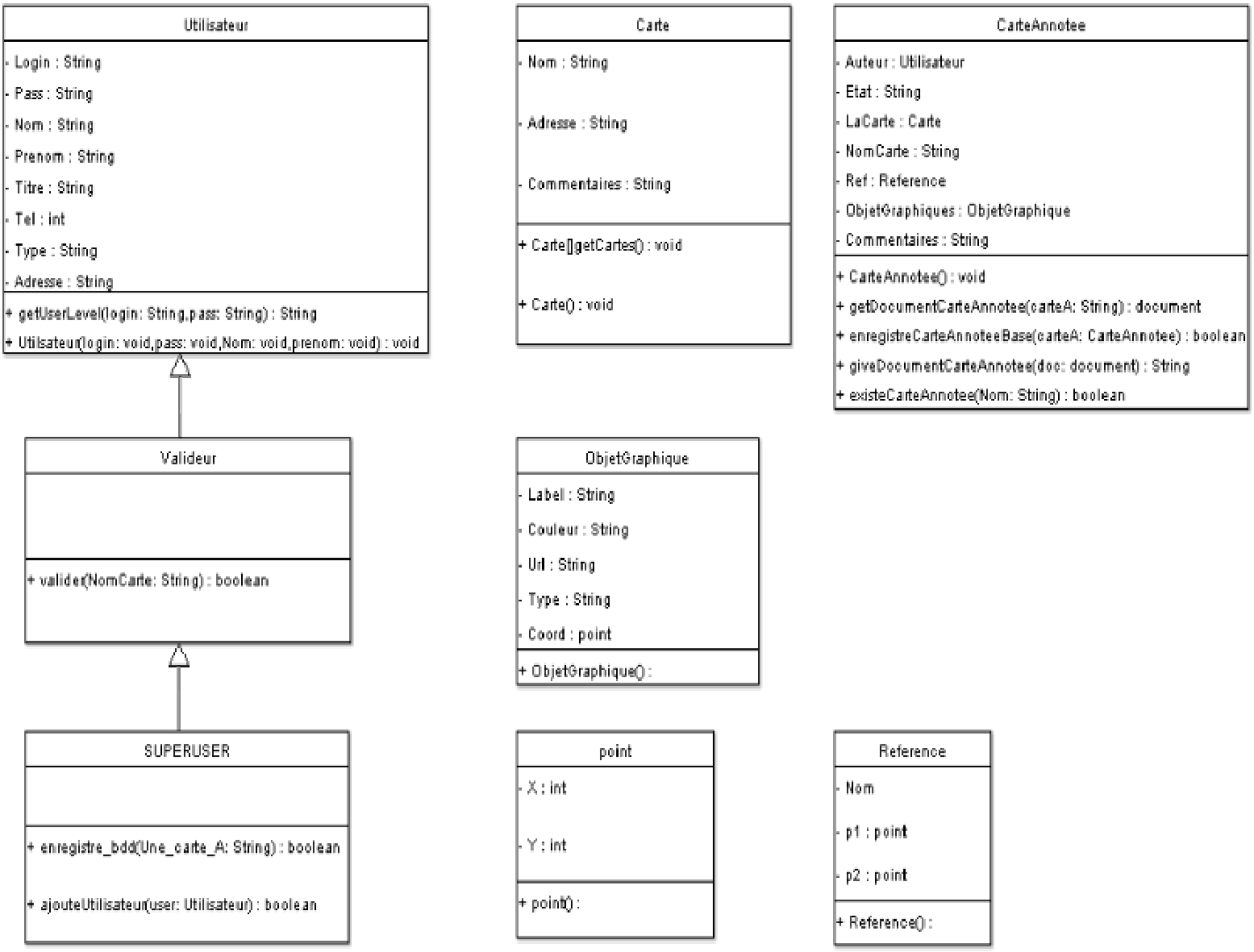
Les tables construites à partir du schéma UML :

```
--table utilisateur
CREATE TABLE utilisateur
(
  login varchar(10) NOT NULL,
  pass varchar(15),
  typeu varchar(10),
  nom varchar(15),
  prenom varchar(15),
  titre varchar(15),
  adresse varchar(50),
  tel int4,
  CONSTRAINT utilisateur_pkey PRIMARY KEY (login),
  CONSTRAINT check_utilisateur_typeu CHECK (typeu in ( 'Valideur', 'SuperUser', 'User'))
)
WITHOUT OIDS;

--table point
CREATE TABLE point
(
  num int4 NOT NULL,
  x int4,
  y int4,
  CONSTRAINT point_pkey PRIMARY KEY (num)
)
WITHOUT OIDS;

--table carte
CREATE TABLE carte
(
  nom varchar(20) NOT NULL,
  commentaire varchar(50),
  url varchar(50),
  CONSTRAINT carte_pkey PRIMARY KEY (nom),
  CONSTRAINT carte_url_key UNIQUE (url)
)
WITHOUT OIDS;
```

```
--table reference
CREATE TABLE reference
(
  nom varchar(20) NOT NULL,
  pointl int4,
  point2 int4,
  CONSTRAINT reference_pkey PRIMARY KEY (nom),
  CONSTRAINT fk_point_referencel FOREIGN KEY (pointl) REFERENCES point (num)
             ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_point_reference2 FOREIGN KEY (point2) REFERENCES point (num)
             ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT check_reference_point CHECK (pointl <> point2)
)
WITHOUT OIDS;


--table objetgraphique
CREATE TABLE objetgraphique
(
  url varchar(30) NOT NULL,
  label varchar(30),
  couleur varchar(20),
  point int4,
  typeo varchar(20),
  commentaires varchar(50),
  CONSTRAINT objetgraphique_pkey PRIMARY KEY (url),
  CONSTRAINT fk_point_objet FOREIGN KEY (point) REFERENCES point (num)
  ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITHOUT OIDS;
--table carteannootee
CREATE TABLE carteannotee
(
nom varchar(20) NOT NULL,
carte varchar(20),
reference varchar(20),
objetgraphique varchar(30),
auteur varchar(10),
etat varchar(10),
commentaire varchar(50),
CONSTRAINT carteannotee_pkey PRIMARY KEY (nom),
CONSTRAINT fk_carteannote_carte FOREIGN KEY (carte) REFERENCES carte (nom)
           ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT fk_carteannote_objetgraphique FOREIGN KEY (objetgraphique)
           REFERENCES objetgraphique (url)
           ON UPDATE NO ACTION
           ON DELETE NO ACTION,
CONSTRAINT fk_carteannote_reference FOREIGN KEY (reference)
           REFERENCES reference (nom)
           ON UPDATE NO ACTION
           ON DELETE NO ACTION,
CONSTRAINT fk_carteannote_utilisateur FOREIGN KEY (auteur)
           REFERENCES utilisateur (login)
           ON UPDATE NO ACTION
           ON DELETE NO ACTION,
CONSTRAINT check_carteannotee_etat CHECK (etat in ('Valide', 'NonValide'))
)
WITHOUT OIDS;
```

Diagramme de classes

Le diagramme de classes déduit est :



Style de codage

Java en général

Les fonctions écrites en Java, elles respectent donc les conventions et le style de codage Java.

Les classes seront précédées de commentaires type *JavaDoc*, et chaque fonction sera décrite (type de valeur de retour, type des paramètres et signification).

Ex :

```
/**
 * Ce servlet reçoit un identifiant de carte et
 * renvoit un document XML a l'applet
 */
public class demande_Annotations extends HttpServlet
{
   /**
   * Constructeur de base
   */
   public demande_Annotations(){
   }

   /** Gère la requête HTTP de type POST
    * @param request servlet requête
    * @param response servlet réponse
    */
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
```

Beans

Les Beans sont utilisés dans les pages JSP de la manière suivantes :

- Déclaration

<jsp:useBean id="monBean" scope="session"
class="enst.cartospip.Beans_identificationUtilisateur" />
ou
<jsp:useBean id="monBean" scope="session" class="enst.cartospip.Beans_recherche" />

- Utilisation des « set » (par exemple setLogin et setPassword à la fois) :

<jsp:setProperty name="monBean" property="*"/>

- Récupération du résultat d'un traitement :

<% ! Carte[] resultat = nomBean.getCartes %>

Servlet

Les deux servlets étendent la classe *HttpServlet*. Ils définissent les méthodes :

- public void init(ServletConfig config) / destroy()
  Initialisation du servlet/Destruction du servlet

- protected void doGet(HttpServletRequest request, HttpServletResponse response)
  Gère les flux HTTP de type GET, ne sera jamais utilisé dans le projet.

- protected void doPost(HttpServletRequest request, HttpServletResponse response)
  Gère les flux http de type POST. Ce sont ces méthodes qui vont réceptionner les demandes d'annotations.

Nous ajouterons la méthode :

- protected void repondre(HttpServletResponse response)
  Cette méthode va répondre au client (Applet), il enverra un document XML (demande d'annotations) ou une chaîne de caractères (confirmation d'ajouts d'annotations).

Versionning

CartoSpip V0.1

Style de codage : indenté (xml), commentaire, génération de java Doc

Description : JSP interagit avec le serveur par l'intermédiaire des Beans (set get anwser), ces dernières permettant d'avoir des réponses à des requêtes du type Afficher carte,..etc

CartoSpip est une interface web pour l'annotation de cartes géographiques en ligne.
CartoSpip est un moyen d'éditer ces cartes par l'ajout de drapeau ou jalons et autres éléments graphiques associé à un commentaire.

Le système offrant ces possibilités est développé de tel façon qu'il peut être intégré à des sites web au profit de communautés souhaitant en faire usage, indépendamment de la plateforme utilisé, la portabilité du système étant assuré grâce à des technologies universelles : JAVA BEANS/APPLET, JSP/Servlet.

CartoSpip respecte les spécifications de l'OpenGis consortium, ce qui permettra son évolutivité.

Le codage et choix des désignations dans la codification sont pensé de façon à facilité sa compréhension de ce code et permettre sa relecture par d'autres développeurs qui souhaiteraient étendre et améliorer les fonctionnalité offerte par la présente version.

Actuellement, ceci est une **version Beta 1**, certaines fonctions pourraient ne pas fonctionné ou n'ont pas encore étaient implémentées.

Le fichier source v0.1 (Java, Beans, XML, JSP etc) sont tous disponible, les versions avenir le seront aussi, de façon à retracer l'évolution du code au fur et à mesure que ce code est enrichi par des fonctionnalité ou renforcé par des corrections.

Nous utilisons actuellement NetBeans IDE 4.1 Early Acces 1 avec JAVA 1.4.2 pour compiler les fichiers sources précités.

Télécharger Les codes sources – Versions – Objet de la modification

Graphic Applet	ver. 0.1	Ajout d'une fonctionnalité pour le déplacement sur la carte - correction du champ d'ajout de lien

Comment annoté une carte avec cartoSpip

-Connectez-vous au site CartoSpip
-Identifiez-vous (il vous faut vous enregistrer pour pouvoir annoter les cartes)
-Après vous être authentifié en tant qu'utilisateur enregistré, vous avez accès sur la page d'accueil par le biais du menu, à la page recherche.

Cette dernière donne accès à un champ de recherche, une fois le choix de la carte ayant été effectué grâce à ce dernier, vous aurez le résultat de votre recherche, a savoir la carte géographique qui vous intéresse dans un applet java qui vous permet d'apporter des ajouts d'éléments graphiques ou textuels de façon à annoter votre carte que vous pouvez sauvegarder, elle sera de ce fait ajouté à la liste des cartes annotées, dans l'attente d'être validée par le Super User

Liste des tâches à faire :

o Compilation et articulation des trois composants de code (interface, applet et base de donnée) pour un premier test transversal de cartospip et l'interaction entre ses derniers
--interface (client) : pages JSP
--applet (client/serveur) : module d'annotation
--base de données (serveur) : corrélation entre les coordonnées géographique de la base de donnée et ceux de la page recherche et de la page de l'applet.

O Implantation de fonctionnalités élémentaire sur les trois composant de code une fois compilé.
-- Interaction entre l'applet, la carte et la base de donnée

-- et entre les résultats de recherche, les cartes et la base de donnée géographique

Liste des tâches effectuées :

o construction graphique des pages du site cartSpip

--FAIT à nouveau dans la version v0.1

o codage préliminaire de l'applet.
-- FAIT dans la version antérieure à v0.1

o installation de la base de donnée PostGres sur le serveur de l'ENST.
-- FAIT dans la version antérieure à v0.1
la Version Tomcat 5 remplacé la version 4 désormais.
Codage du fichier XML pour l'interaction Client/serveur/base données

Tâches hors-list :

1. Perfectionnement des éléments graphiques (secondaire)
2. Utilisation de MySQL (n'intègre pas certains éléments nécessaires à notre projet)
3. Exportation des cartes sur mobile/wap (pas à l'ordre du jours, délais de développement restreint)

Bugs rencontré :

1. L'applet n'affichait pas correctement les champs de saisi des commentaires et liens, du à l'oubli d'une class qui n'avais pas été placé sur le serveur, le problème a été résolu dans la présente version.
2. Bug suite à des restrictions de lecture sur l'applet (module d'affichage)

Problèmes éventuels :

L'applet pourrait ne pas fonctionner :

→ sur des navigateurs IE et NS antérieures aux versions 4x
→ sur les navigateurs ne disposants pas de versions ultérieures au J2E 1.4

Mise à jour de la planification :

Equipe	Tâche	Durée
Benaissa	Regroupements des modules Tests du module d'affichage web JSP	Du 30/11/04 Au 05/12/04 Du 05/12/04 Au 15/12/04
Pajaniaye	Regroupements des modules Tests du module interaction serveur/base de données	Du 30/11/04 Au 05/12/04 Du 05/12/04 Au 15/12/04
Rabiaza	Regroupements des modules Tests du module interaction serveur/client	Du 30/11/04 Au 05/12/04 Du 05/12/04 Au 15/12/04

Conclusion

Dans cette revue nous avons décrit l'organisation des composants et leurs interactions, mais aussi la structuration de la documentation élaborée pour retracer les évolutions et modifications marquant le codage puis les tests à venir. Au niveau gestion de projet, nous pensons êtres dans le respect du planning présenté lors de la première revue, bien que l'ampleur des taches semble plus large que celle que nous avons anticipé.

Cette partie nous a également permis de valider auprès de l'expert la structure du XML et les fonctionnalités offertes par la plateforme CartoSpip.