Rapport de stage

Effectué à l'IRD en Guinée
Du 10 juillet au 24 août 2001

Système d'information dynamique appliqué aux pêches maritimes guinéennes

Misako ITO
Filière Electronique 2ème année
Année 2000-2001

INP ENSEEIHT

Remerciements

Je voudrais particulièrement remercier mon maître de stage, M. Loïc THIBAUT, pour m'avoir proposé un sujet de stage aussi passionnant, m'avoir fait partager ses convictions pour l'Afrique, le développement et les systèmes d'information et m'avoir assistée avec beaucoup d'attention tout au long de mon stage.

Je remercie également le Directeur M. André FONTANA ainsi que toute l'équipe des chercheurs de l'IRD et du CNSHB pour leur accueil très chaleureux.

Enfin un grand remerciement à l'équipe informatique du CNSHB pour leur gentillesse et leur disponibilité, notamment à Mamadi FOFANA et à Moustafa SYLLA grâce à qui j'ai passé des moments très agréables.

Sommaire

I. Présentation

1. L'IRD et le contexte du développement en Guinée

L'IRD est un établissement public à caractère scientifique et technologique qui, placé sous la tutelle des ministres chargés de la Recherche et de la Coopération, conduit des recherches sur les milieux intertropicaux depuis cinquante ans. Ses recherches scientifiques sont centrées sur les relations entre l'homme et son environnement dans les régions tropicales et méditerranéenne, dans la perspective d'un développement durable de ces régions. L'Institut est présent en France métropolitaine, dans les territoires et départements d'Outre mer et dans plus de 40 pays du monde tropical dont la Guinée où j'ai effectuée mon stage.

La Guinée est une république démocratique située en Afrique de l'Ouest. Ancienne colonie française indépendante depuis 1958, après une longue période de dictature sous Sékou Touré, le pays se mobilise dès 1984 pour donner un nouvel élan vers la voie démocratique en s'ouvrant au monde et en instaurant le multipartisme et les élections libres. Economiquement, la Guinée joue un rôle actif dans la production des minerais avec ses réserves de bauxite qui sont parmi les plus riches du monde et ses gisements de fer, de diamant, d'or et de l'uranium.

Toutefois, les risques de marginalisation subsistent toujours face à la globalisation de l'économie et la Guinée demeure un Etat de droit au service du développement.

Population	8 000 000 habitants Conakry : 1 000 000 habitants	Moins de 15 ans : 44 % Plus de 65 ans : 3 %
Superficie	248 547 km^2	
Densité de population	31 habitants / km²	Densité urbaine : 31 %
Croissance démographique	2.9 % / an	
Fécondité	6.2 enfants / femme	
Mortalité	20 décès / 1000 habitants	Mortalité infantile : 130 décès / 1000 habitants
Espérance de vie	45 ans	
Alphabétisation	24 %	Hommes : 35 % Femmes : 13 %
Religion	Musulmans : 90 % Chrétiens : 5 % Animistes : 5 %	

Figure 1 Quelques chiffres indicateurs du développement

2. Le CNSHB et le projet PEG

Dans ce contexte, l'IRD mène des projets de recherche en coopération avec des institutions locales en y apportant un fort appui technique, et c'est dans une de ces institutions, le CNSHB (Centre National des Sciences Halieutiques de Boussoura), que j'ai travaillé pendant mon séjour à Conakry. Le CNSHB est un établissement public à caractère scientifique, placé sous la tutelle du Ministère chargé de la pêche. Sa mission est de contribuer au développement de la pêche maritime en Guinée par une meilleure connaissance

et évaluation des ressources halieutiques. En effet, la pêche artisanale représente une des principales activités du pays : elle est pratiquée par environ 10000 pêcheurs principalement au moyen des pirogues dont la moitié sont équipées de moteur, et chaque année plusieurs dizaines de tonnes sont pêchées dont quelques milliers exportées.

Dans ce cadre, le projet PEG (Pêche Ecologique Guinéenne) qui a débuté en septembre 2000 et qui est mené par l'équipe de l'IRD et du CNSHB a pour objectif d'établir les modalités d'un développement durable de la pêche maritime guinéenne. Ce projet est fondé sur l'usage respectueux des écosystèmes marins afin d'éviter la surexploitation et donc l'extinction des ressources halieutiques guinéennes. Il consiste notamment à préserver les capacités productives de l'écosystème et à développer au sein du secteur une connaissance sur les effets de l'activité humaine.

Les principaux volets d'activité de ce projet sont l'acquisition de la connaissance sur le secteur de la pêche et la modélisation de l'écosystème marin et de l'exploitation guinéenne, qui sera restituée sous forme d'un logiciel de simulation. Le modèle adopté est un modèle multi-agents où chaque information est représentée sous forme d'un agent dans un environnement Java, et cette modélisation sera effectuée à partir des informations qui seront rassemblées dans un centre d'information.

3. L'objectif du stage

L'objectif de mon stage est la construction de ce centre d'information qui va non seulement permettre de modéliser la circulation de l'information au sein du secteur mais aussi constituer un véritable système d'information sur la pêche maritime guinéenne. Dans cette perspective, le centre doit rassembler et restituer l'ensemble des connaissances et des données relatives au projet sous forme d'un site Internet.

II. Centre d'information

1. Le cahier des charges

Le centre d'information s'adressera aussi bien aux chercheurs qu'aux différents acteurs de la pêche (pêcheurs, mareyeurs, gestionnaires…) donc à des utilisateurs de profils très différents. De même, les informations qui composent le centre sont très hétérogènes par leur nature, leur forme, leur mode de représentation et leur complexité. Il doit par conséquent :

- disposer d'une architecture générique pour héberger les informations,
- permettre l'intégration de toute nouvelle information et
- permettre la restitution de l'information sous une forme adaptée.

Il se présentera donc sous forme d'un site Web dynamique relié à une base de données Access alimentée par un comité de lecture. Un formulaire d'entrée des données a été crée afin de remettre le pouvoir de publication entre les mains du créateur de contenu et non d'un technicien et de pouvoir ainsi récolter un maximum d'informations. La base stocke des pointeurs sur des fichiers qui contiennent l'information proprement dite ainsi qu'une série de descripteurs (dimension, source, mots-clés...) permettant de caractériser l'information. Chaque information de la base est identifiée par sa clé primaire qui est son numéro ID.

L'extraction des données contenues dans le base et la génération des pages HTML correspondantes se feront dans un environnement Java conformément au choix de l'environnement qui a été fait pour la modélisation. On utilisera pour cela les technologies XML et XSLT suivant le schéma ci-dessous :

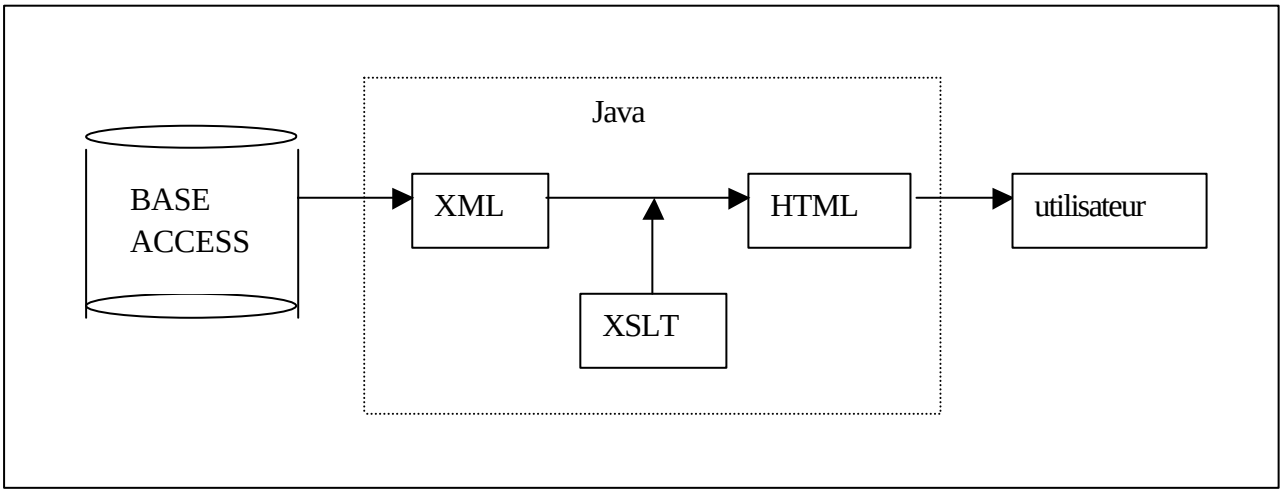


Figure 2 Schéma de génération de l'information

2. Les choix technologiques

Une des préoccupations majeures pour concevoir un système d'information est son interopérabilité avec d'autres applications qui peuvent évoluer de manière très distincte. Les choix technologiques doivent donc répondre à ce besoin d'interopérabilité. C'est le cas du

couple Java-XML/XSLT qui satisfait les critères de facilité d'utilisation, d'évolutivité et d'ouverture aux standards Web.

a. Le XML

Le langage XML (Extensible Markup language) issu du SGML (Structured Generalized Markup Language) est un métalangage de définition de types de documents. Dans ce sens, il permet de fixer les règles de syntaxe de tout nouveau type de documents tout comme le SGML qui a défini le type de document HTML.

Le XML a l'avantage d'abolir les limites du HTML tout en évitant la trop grande complexité du SGML qui a freiné son développement. En effet, alors que le texte d'un document HTML repose sur des données non structurées qui ne peuvent être traitées par un programme, un document XML présente une structure intermédiaire entre celle rigoureuse d'une base de données et celle inexistante d'un texte. Il fournit ainsi des données aptes à être traitées.

Cette extensibilité du XML s'explique par l'introduction d'un nouveau type de balises appelées des balises sémantiques. Celles-ci ne donnent ni indication de structure ni celle de formatage mais sont accessibles aux scripts et aux programmes. Elles permettent ainsi l'application d'autres technologies telles que le HTML dynamique ou les feuilles de style. De cette manière, le XML accroît les possibilités de créer des descriptions adaptées aux besoins et de générer des pages plus interactives.

b. Le XSLT

L'idée d'écarter les balises de formatage dans un document XML a consolidé son association avec le langage XSL (Extensible Stylesheet Language) qui permet gérer l'affichage des données. Le rôle du langage XSLT (XSL Transforms) est alors de transformer des documents en appliquant les feuilles de style XSL. Dans notre cas le programme XSLT contiendra des instructions de transformation d'un document XML en un document HTML.

Le document XML généré à partir de la base de données et contenant l'information sera le résultat d'un programme. Celui-ci sera envoyé à un processeur XSLT afin d'être converti en document HTML selon les instructions d'un document XSLT associé. Lorsque le document HTML est reçu par le navigateur, celui-ci ne sait pas que le document original est un document XML. Dans ce modèle, la transformation se fait du côté serveur et le navigateur affiche simplement le document HTML.

On aurait pu envoyer le document XML accompagné du document XSLT au navigateur, ce qui lui présente l'avantage de posséder en mémoire les deux documents XML et HTML et donc de pouvoir transformer le document XML plusieurs fois. Toutefois tous les navigateurs ne sont pas forcément capables de réaliser la transformation et ne possèdent pas un processeur XSLT. Notre modèle utilisera donc juste la fonction principale du navigateur qui est d'interpréter HTML.

3. Les maquettes HTML

Les réponses à la requête sous forme de page HTML auront deux styles distincts selon qu'il s'agit d'envoyer une liste d'informations ou l'information proprement dite.

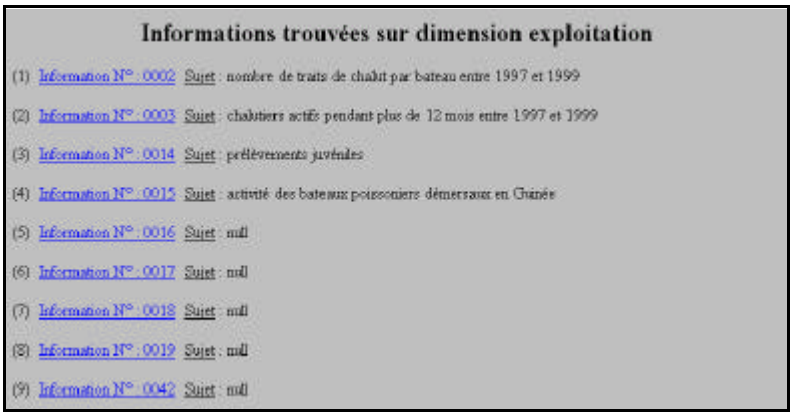


Figure 3 Maquette HTML représentant une liste d'informations

Figure 4 Maquette HTML représentant une information

Le navigateur affichera les fichiers HTML dans trois frames, un contenant la réponse de la requête, un deuxième contenant un fichier de dimensions et un autre contenant un fichier de logos qui pointeront vers les sites partenaires du CNSHB.

Figure 5 Maquette d'affichage de la requête

III. Modélisation

1. L'approche objet

Un objet est une représentation abstraite d'un monde et possède un sens précis dans le contexte du problème. Il est par conséquent défini par un état caractérisé par les valeurs de ses attributs et des comportements qui sont décrits par ses méthodes.

La modélisation objet a pour but de décrire ces objets en représentant leurs relations dans le système. Cette approche de la modélisation prend son intérêt par rapport à programmation traditionnelle dans la mesure où elle correspond étroitement au monde réel et permet la décomposition du problème en sous-problèmes. Le modèle objet est ainsi plus portable et plus souple pour les modifications.

Nous avons ainsi structuré l'application qui va de l'extraction des données à la génération des pages HTML en trois couches. Les couches représentent un ensemble de classes en Java et se communiquent entre-elles à travers des API (*Application Programming Interface*) qui sont des méthodes publiques d'une classe. En effet, dans la modélisation objet, on distingue la classe d'objet qui décrit un groupe d'objets ayant les mêmes propriétés et le même comportement et l'instance d'objet qui représente un objet particulier.

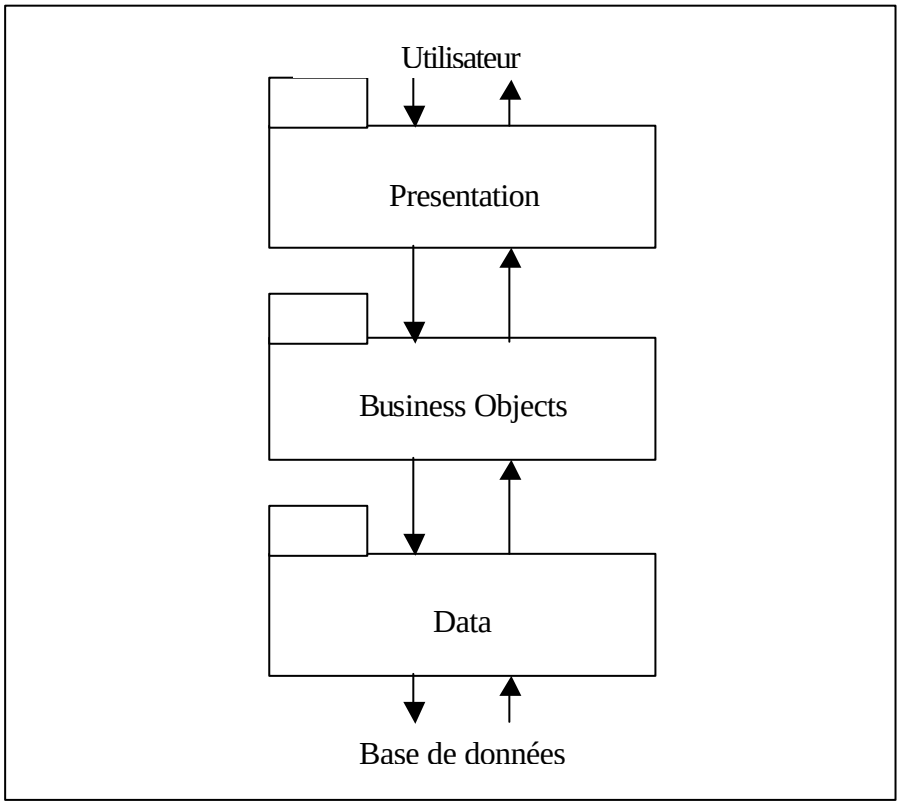


Figure 6 Modélisation du centre d'informations

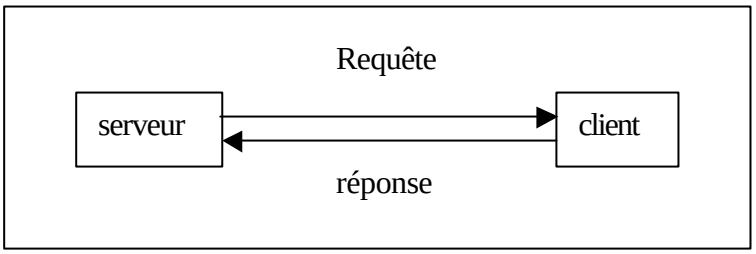
a. La couche Presentation

Elle représente l'interface avec l'utilisateur. Elle est composée de deux sous-couches :

-une permettant la création des documents XML à partir des objets générés par la couche Business Objects,

-une permettant la transformation d'un document XML en HTML en appliquant la feuille de style XSLT et l'envoi au navigateur des documents HTML ainsi générés à l'aide des Servlets.

Un Servlet est un programme en Java qui, exécuté au sein d'un serveur Web, permet de gérer la relation requête-réponse du client et du serveur :



On définit dans cette couche deux Servlets : un qui sera exécuté lorsque l'utilisateur cliquera sur un lien vers une information et un autre pour un lien vers une liste d'informations.

b. La couche Business Objects

C'est dans cette couche que sont définis et créés les objets Java qui seront manipulés par le programme. Les principaux objets manipulés sont instanciés par les classes :

-Information qui comporte des variables d'instances privées représentant tous les descripteurs de l'information et des méthodes publiques permettant d'y accéder,

-ListeInformations qui comporte une collection d'informations ayant en commun les paramètres du descripteur et qui possède en méthode publique un itérateur permettant d'accéder à cette liste d'informations,

-DescripteurSimple et DescripteurComposite qui implémentent l'interface Descripteur.

Un descripteur est dit simple lorsqu'à un type du descripteur on peut associer un contenu unique (ex : support, dimension, date_info…) et il est qualifié de composite lorsqu'à un type du descripteur on peut associer plusieurs contenus (ex : mots_cles, composants).

Un objet descripteur comporte donc en attributs privés une chaîne de caractères représentant le type du descripteur et une chaîne de caractères ou une collection représentant le ou les contenus du descripteur selon qu'il soit simple ou composite.

c. La couche Data

C'est dans cette couche que s'effectue la connexion avec la base de données et l'extraction de l'information requêtée sous un format générique. Cette information est ensuite envoyée à la couche supérieure Business Objects qui crée l'objet approprié.

Les interactions entre les classes de ces trois couches représentent la structure statique du système à concevoir.

2. Le diagramme de classes

Le diagramme de classes défini par UML (*Unified Modeling Language*), le nouveau concept de modélisation objet, permet de représenter cette structure du système en terme de classes et de relations. L'édition du diagramme a été effectuée à l'aide du logiciel Rational Rose qui, relié à un éditeur Java, permet la génération automatique des classes.

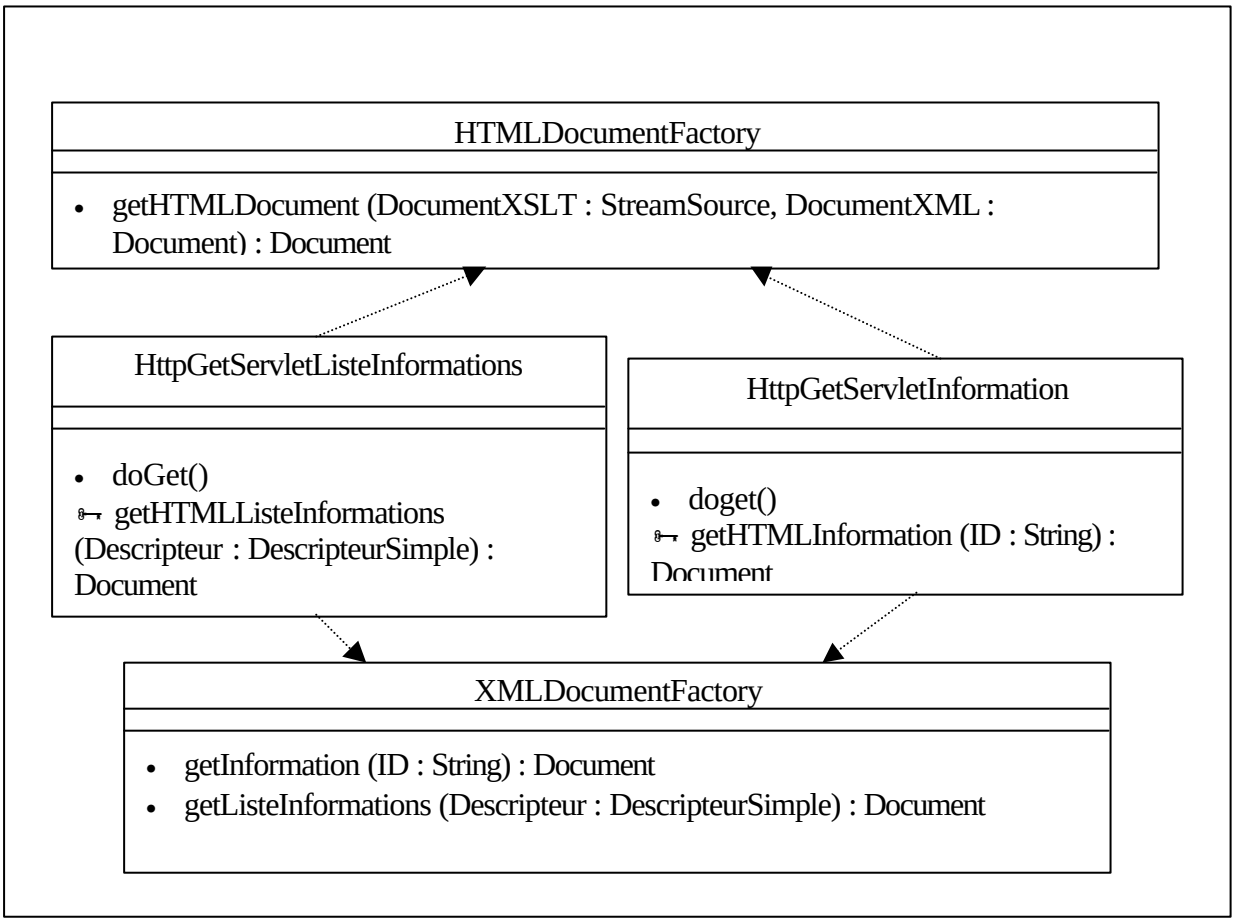


Figure 7 Diagramme de classes de la couche Presentation

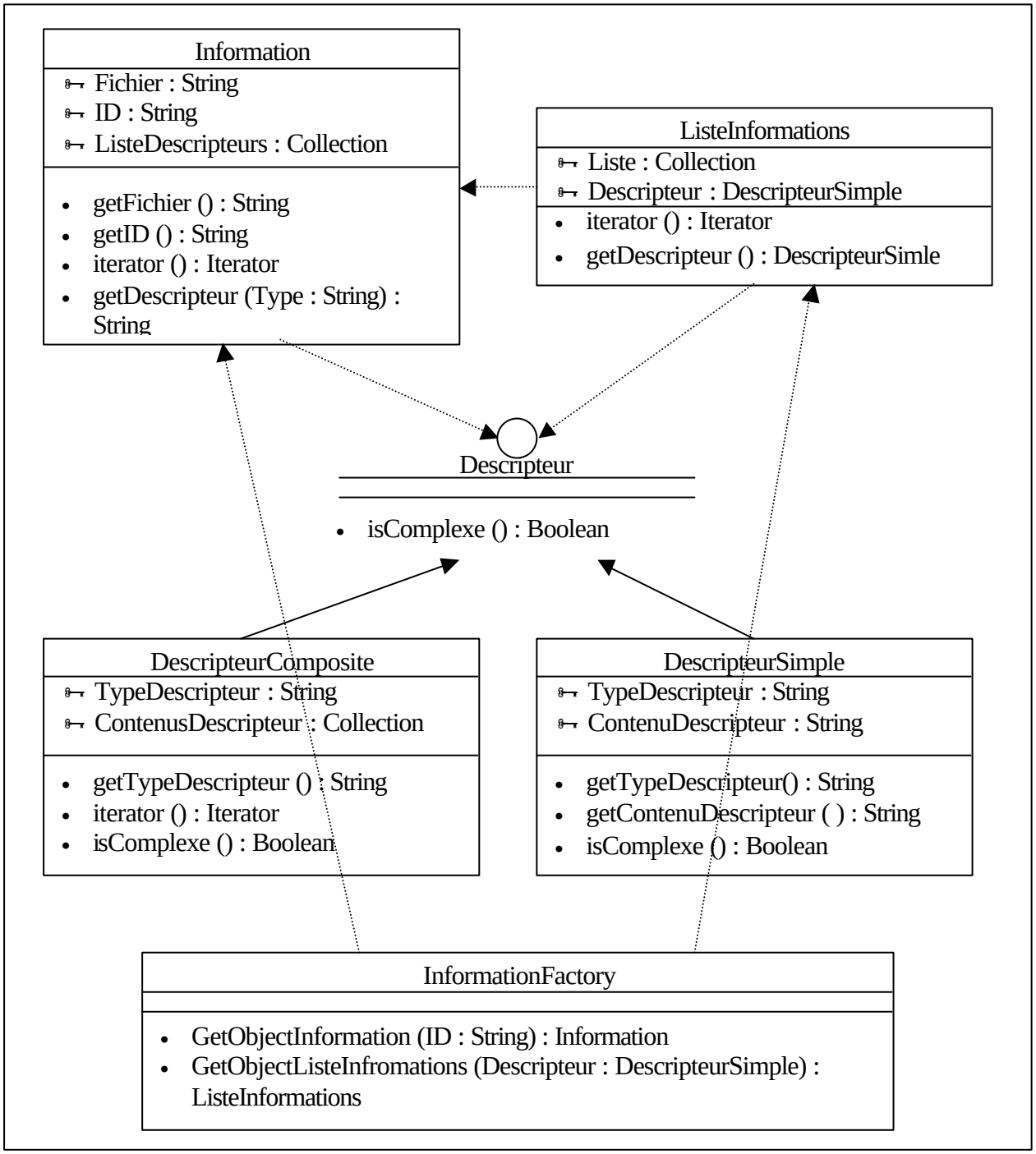


Figure 8 Diagramme de classes de la couche Business Objects

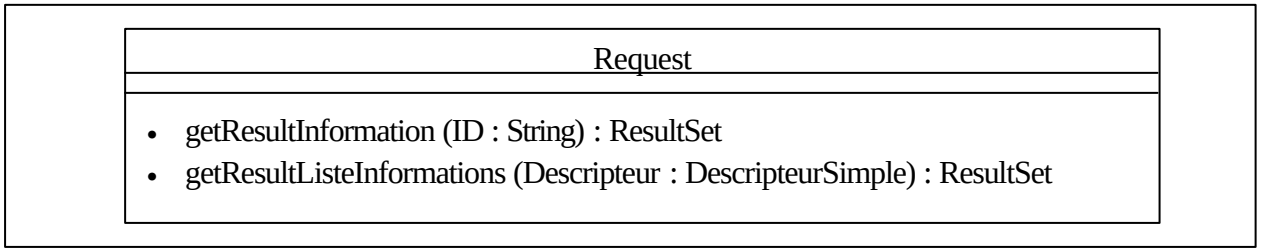


Figure 9 Diagramme de classes de la couche Data

3. Les modèles XML

Les modèles des documents XML que la classe XMLDocument Factory doit générer sont les suivants :

- pour un document XML présentant une information

```
<?xml version="1.0" encoding="UTF-8" ?>
   <information id="0005" fichier="0005.xls">
      <titre>Débarquements (en t.) de la pêche maritime en ZEE guinéenne</titre>
      <support>statistique</support>
      <dimension>Production</dimension>
      <date_info>1995</date_info>
      <lieu_info>ZEE</lieu_info>
      <composants>
         <composant>bateau</composant>
         <composant>pêcheur</composant>
         <composant>oiseau</composant>
      </composants>
      <mots_clés>
         <mot_clé>statistiques</mot_clé>
         <mot_clé>débarquements</mot_clé>
      </mots_clés>
      <niveau_confiance>inconnu</niveau_confiance>
      <source_info>CNSHB</source_info>
  </information>
```

On définit ainsi sous XML une information par un élément « information » comportant des attributs et des sous-éléments qui sont les descripteurs. L'attribut «id » permet d'identifier de manière unique l'information et « fichier » indique le nom du fichier contenant l'information. Parmi les sous-éléments qui composent l'information, on peut distinguer par la structure du document XML, les descripteurs simples et les descripteurs composites. Les descripteurs composites sont représentés sous XML par des éléments comportant des sous-éléments dont le nom est identique à celui de l'élément père au singulier et dont le nombre correspond au nombre de contenus.
Cette structure du document XML permet de référencer chaque composant d'une information pour y faciliter l'accès.

-pour un document XML présentant une liste d'informations

```
<?xml version="1.0" encoding="UTF-8" ?>
<liste type="dimension" contenu="exploitation">
   <information id="0002">
      <sujet>nombre de traits de chalut par bateau entre 1997 et 1999</sujet>
   </information>
   <information id="0003">
      <sujet>chalutiers actifs pendant 12 mois entre 1997 et 1999</sujet>
   </information>
   <information id="0015">
      <sujet>null</sujet>
   </information>
</liste>
```

Une liste d'informations est définie par ses attributs et rassemble toutes les informations qui ont le même descripteur c'est à dire même «contenu » pour un même «type » du descripteur. Chaque information faisant partie de cette liste possède un attribut « id » et un élément-fils « sujet ».

IV. Réalisation

1. Le codage

a. Java

Pour le codage des classes Java ainsi modélisées, j'ai travaillé en binôme avec Mamadi FOFANA, un des membres de l'équipe technique du CNSHB. Je me suis chargée du codage de la couche Presentation et Mamadi des couches Business Objects et Data. Les codes sources des différentes classes sont disponibles en Annexes du rapport.

b. XSLT

Je me suis également occupée de la création des feuilles de style XSLT qui permettent la mise en page des éléments du document XML conformément à sa maquette HTML. J'ai donc crée deux feuilles de style, une pour la mise en page d'une information et une autre pour celle d'une liste d'informations. Les sources de ces documents sont également disponibles en Annexes.

2. Les tests

Les tests de validation ont été effectués pour chaque couche par intégration successive des classes dans la maquette finale. Nous avons ainsi vérifié le fonctionnement du centre en suivant les étapes :

-extraction de la base de l'information correspondant à la requête,
-génération du document XML correspondant et
-transformation du document XML en document HTML correspondant.

3. L'intégration

Afin de faciliter la correction des erreurs provenant de différentes couches, nous les avons fait propager vers la couche supérieure pour les faire afficher à l'écran lors de l'exécution du programme.

V. Bilan

1. Les résultats

Nous avons ainsi réalisé une première version du centre d'informations qui marche !! Lorsque l'utilisateur fait une requête, la page HTML générée affiche l'information demandée dans la même fenêtre sauf s'il s'agit d'un document Excel, auquel cas l'information est affichée dans une fenêtre Excel indépendante.

2. Les améliorations possibles

On pourra apporter dans les versions futurs de ce centre d'informations des améliorations au niveau de la présentation, de la structuration du code et d'optimisation du temps de réponse.

En effet, on pourra non seulement améliorer la mise en page et le graphisme des documents HTML générés, mais aussi l'affichage de l'information en prenant en compte la spécificité en formatage de chacune.

Quant au code, on pourra le structurer de manière à le rendre encore plus transparent et faciliter sa mise à jour tout en conservant la modélisation initiale en trois couches qui étaient prévues à cet effet.

Enfin, on pourra également chercher à optimiser le temps de réponse du site en restructurant certaines classes telles que ListeInformations qui comportent une collection d'informations avec tous ses descripteurs alors que seuls les descripteurs « ID » et « sujet » sont nécessaires à l'affichage de la page.

3. Les perspectives du projet PEG

La création de cette première version du centre d'informations permet le démarrage du deuxième volet du projet PEG, c'est à dire la modélisation, dont la fin est prévue pour septembre 2003. Le projet est certes ambitieux mais dessert des macro-objectifs du développement en instaurant les nouvelles technologies comme levier économique et social.

En effet, au delà de sa fonctionnalité, le centre d'informations ainsi réalisé a pour but de promouvoir l'usage du système d'informations comme facteur du développement du secteur de la pêche en Guinée. Il n'est pas une fin en soi, mais un moyen qui s'inscrit dans un processus plus général de renforcement des organisations et des initiatives locales de ce secteur.

Accompagnée d'une volonté politique et d'un soutien financier, la technologie de l'information peut ainsi donner un élan accélérateur au développement du pays et apporteraient des solutions du troisième millénaire aux problèmes des pays du Sud.

Annexes

1. Les codes sources de la couche Presentation

a. La classe XMLDocumentFactory

```
//Source file: C:\\peg\\ci\\xml\\XMLDocumentFactory.java

package peg.ci.xml;

import peg.ci.bo.*;
import java.io.*;
import java.util.*;
import org.w3c.dom.*;
import javax.xml.parsers.*;

public class XMLDocumentFactory extends Object {

   /** Creates new XMLDocumentFactory */
   public XMLDocumentFactory() {
   }

   public Document GetInformation (String ID)
     throws DOMException, ParserConfigurationException {

      //instanciation des objets
      InformationFactory IF = new InformationFactory();
      Information info = IF.GetObjectInformation(ID);
      Iterator i = info.Iterator();

      //instanciation du document
      DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
      DocumentBuilder db = dbf.newDocumentBuilder();
      Document doc = db.newDocument();

      //création de la balise information avec son attribut et de la balise fichier
      Element information = doc.createElement("information");
      information.setAttribute("id",ID);
      doc.appendChild(information);
      Element fichier = doc.createElement("fichier");
      fichier.appendChild(doc.createTextNode(i.GetFichier()));
      information.appendChild(fichier);

      //création des autres balises descripteurs
      while ( i.hasNext()){
         Descripteur d = i.next();
         if (d.isComplexe()){
            while (d.Iterator().hasNext()){
               Element elem = doc.createElement(d.GetTypeDescripteur());
               elem.appendChild(doc.createTextNode(d.Iterator().next()));
               information.appendChild(elem);
               }
            }
         else {
            Element elem = doc.createElement(d.GetTypeDescripteur());
            elem.appendChild(doc.createTextNode(d.GetContenuDescripteur()));
            information.appendChild(elem);
            }
      }
      return doc;
   }

   public Document GetListeInformations (DescripteurSimple descripteur)
      throws DOMException, ParserConfigurationException {

      //instanciation des objets
```

```
        InformationFactory IF = new InformationFactory();
        ListeInformations li = new ListeInformations();
        li = IF.GetObjectListeInformations(descripteur);
        Iterator i = li.Iterator();

        //instanciation du document
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        Document doc = db.newDocument();

        //création de la balise liste avec ses 2 attributs
        Element liste = doc.createElement("liste");
        liste.setAttribute("type",descripteur.GetTypeDescripteur());
        liste.setAttribute("contenu",descripteur.GetContenuDescripteur());
        doc.appendChild(liste);

        //création des autres balises
        while ( i.hasNext()){
           Information info=i.next();
           Element information = doc.createElement("information");
           information.setAttribute("id",info.GetID());
           liste.appendChild(information);
           Element sujet = doc.createElement("sujet");
           sujet.appendChild(doc.createTextNode(info.GetDescripteur("sujet")));
           information.appendChild(sujet);
        }
        return doc;
    }

    public Document GetListeDimensions ()
        throws DOMException, ParserConfigurationException {

        //instanciation des objets
        InformationFactory IF = new InformationFactory();
        ListeDimensions ld = new ListeDimensions();
        ld = IF.GetObjectListeDimensions();
        Iterator i = ld.Iterator();

        //instanciation du document
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        Document doc = db.newDocument();

        //création de la balise liste avec ses 2 attributs
        Element liste = doc.createElement("liste");
        doc.appendChild(liste);

        //création des autres balises
        while ( i.hasNext()){
              Element dimension = doc.createElement("dimension");
              dimension.appendChild(doc.createTextNode(i.next()));
              liste.appendChild(dimension);
        }
        return doc;
    }
}
```

b. La classe HTMLDocumentFactory

```
//Source file: C:\\peg\\ci\\presentation\\HTMLDocumentFactory.java

package peg.ci.presentation;

import javax.xml.transform.dom.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import org.xml.sax.*;
import java.io.*;
import java.util.*;
import org.w3c.dom.*;
import javax.xml.parsers.*;

//contient une méthode qui permet de transformer un fichier xml en un fichier html en utilisant
//une feuille de style xslt
```

```
public class HTMLDocumentFactory{

   public HTMLDocumentFactory(){
   }

   public Document GetHTMLDocument(StreamSource documentXSLT, Document documentXML)
     throws TransformerException, TransformerConfigurationException, IOException, SAXException,
ParserConfigurationException{

      //instanciation du document
       DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
       DocumentBuilder db = dbf.newDocumentBuilder();
       Document doc = db.newDocument();

       //instanciation d'un TransformerFactory
       TransformerFactory tFactory = TransformerFactory.newInstance();

       //utilisation du TranformerFactory pour traîter la feuille de style xslt et géné un Transformer
            Transformer transformer = tFactory.newTransformer(documentXSLT);

       //utilisation du Transformer pour transformer un fichier xml et
       //envoyer la sortie dans un fichier html
            transformer.transform(new DOMSource(documentXML), new DOMResult(doc));

       return doc;
       }
}
```

c. La classe HttpGetServletInformation

```
//Source file: C:\\peg\\ci\\presentation\\HttpGetServletInformation.java

package peg.ci.presentation;

import peg.ci.xml.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import org.w3c.dom.*;

public class HttpGetServletInformation extends HttpServlet
{
   public HttpGetServletInformation()
  {

  }

   public void doGet (HttpServletRequest requete, HttpServletResponse reponse)
      throws ServletException, IOException{

      reponse.setContentType("text/html");

      String id = requete.getParameter("id");

      Document doc = GetHTMLInformation(id);

      PrintWriter sortie = reponse.GetWriter();
      sortie.println(doc.toString());
      sortie.close();

  }


  private Document GetHTMLInformation(String ID)
  {
   XMLDocumentFactory df = new XMLDocumentFactory();
   HTMLDocumentFactory sf = new HTMLDocumentFactory();
   Document docXML = df.GetInformation(ID);
   Document docHTML = sf.GetHTMLDocument(new StreamSource("c:\\misako\\xml\\info1.xsl"),docXML);
   return docHTML;
  }
}
```

d. La classe HttpGetServletListeInformations

```
//Source file: C:\\peg\\ci\\presentation\\HttpGetServletListeInformations.java

package peg.ci.presentation;

import peg.ci.xml.*;
import peg.ci.bo.*;
import org.w3c.dom.*;

public class HttpGetServletListeInformations extends HttpServlet
{
  public HttpGetServletListeInformations()
  {
  }

  public void doGet()
  {
     reponse.setContentType("text/html");
     String type = requete.getParameter("type");
     String contenu = requete.getParameter("contenu");
     DescripteurSimple ds = new DescripteurSimple(type, contenu);
     HTMLDocument doc = GetHTMLListeInformations(ds);

     PrintWriter sortie = reponse.GetWriter();
     sortie.println(doc.toString());
     sortie.close();

  }

  private Document GetHTMLListeInformations(DescripteurSimple Descripteur)
  {
   XMLDocumentFactory df = new XMLDocumentFactory();
   HTMLDocumentFactory sf = new HTMLDocumentFactory();
   Document docXML = df.GetListeInformations(Descripteur);
   Document docHTML = sf.GetHTMLDocument(new StreamSource("c:\\misako\\xml\\infos1.xsl"),docXML);
   return docHTML;
  }
}
```

2. Les feuilles de style XSLT

a. Pour l'information

```
<?xml version="1.0" encoding="UTF-8" ?>
 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <xsl:output method="html" indent="yes" doctype-public="-//W3C//DTD HTML 4.0//EN"
      />
   <xsl:template match="/">
      <html>
         <head>
            <title>Centre d'information de CNSHB</title>
         </head>
         <body>
            <xsl:apply-templates />
         </body>
      </html>
   </xsl:template>
   <xsl:template match="information">
      <h4 align="center">TITRE</h4>
      <p align="center">
         <b><xsl:value-of select="titre" /> </b>
      </p>
      <p align="center">
         <img border="1" src=".../infos/{fichier}" width="400" height="200" />
      </p>
```

```
<p>
  Représentation disponible pour cette information :
   <a href="file://localhost/c:/cixml/GetList.html?type=support&contenu={support}">
    <xsl:value-of select="support" />
  </a>
</p>
<p>NAVIGATION :</p>
 <p>
  Dimension :
  <a
     href="file://localhost/c:/cixml/GetList.html?type=dimension&contenu=
     {dimension}">
    <xsl:value-of select="dimension" />
  </a>
</p>
 <p>
  Date :
  <a
     href="file://localhost/c:/cixml/GetList.html?type=date_info&contenu=
     {date_info}">
    <xsl:value-of select="date_info" />
  </a>
</p>
<p>
  Lieu :
  <a
     href="file://localhost/c:/cixml/GetList.html?type=lieu_info&contenu={
     lieu_info}">
    <xsl:value-of select="lieu_info" />
  </a>
</p>
<p>
  Composants :
  <xsl:apply-templates select="composants" />
</p>
<p>
  Mots-clés :
  <xsl:apply-templates select="mots_clés" />
</p>
<p align="center">
  Information N°
  <xsl:value-of select="@id" />
   - Niveau de confiance :
  <a
     href="file://localhost/c:/cixml/GetList.html?type=niveau_confiance&co
     ntenu={niveau_confiance}">
    <xsl:value-of select="niveau_confiance" />
  </a>
   - Source :
  <a href =
     "file://localhost/c:/cixml/GetList.html?type=source_info&contenu={so
     urce_info}">
    <xsl:value-of select="source_info" />
  </a>
</p>
</xsl:template>
<xsl:template match="composants">
  <xsl:for-each select="composant">
    <a
       href="file://localhost/c:/cixml/GetList.html?type=composant&contenu
       ={.}">
      <xsl:value-of select="." />
    </a>
  </xsl:for-each>
</xsl:template>
 <xsl:template match="mots_clés">
   <xsl:for-each select="mot_clé">
```

```
            <a href =
"file://localhost/c:/cixml/GetList.html?type=mot_clé&contenu={.}">
                <xsl:value-of select="." />
            </a>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

b. Pour la liste d'informations

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:output method="html" indent="yes" doctype-public="-//W3C//DTD HTML 4.0//EN"
        />
    <xsl:template match="/">
        <html>
            <head>
                <title>Centre d'information de CNSHB</title>
            </head>
            <body>
                <h2 align="center">
                    Informations trouvées sur
                    <xsl:value-of select="liste/@type" />
                    <xsl:value-of select="liste/@contenu" />
                </h2>
                <xsl:apply-templates select="liste/information" />
            </body>
        </html>
    </xsl:template>
    <xsl:template match="information">
        <p>
            <xsl:number value="position()" format="(1)" />
            <a href="file://localhost/c:/cixml/GetInfo.html?id={@id}">
                Information N° : <xsl:value-of select="@id" />
            </a>
            <u>Sujet</u>
            : <xsl:value-of select="sujet" />
        </p>
    </xsl:template>
</xsl:stylesheet>
```