CartoSpip

TELECOM PARIS
école nationale supérieure des télécommunications

CartoSpip

Web cartographique Coopératif

Analyse des besoins logiciels

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

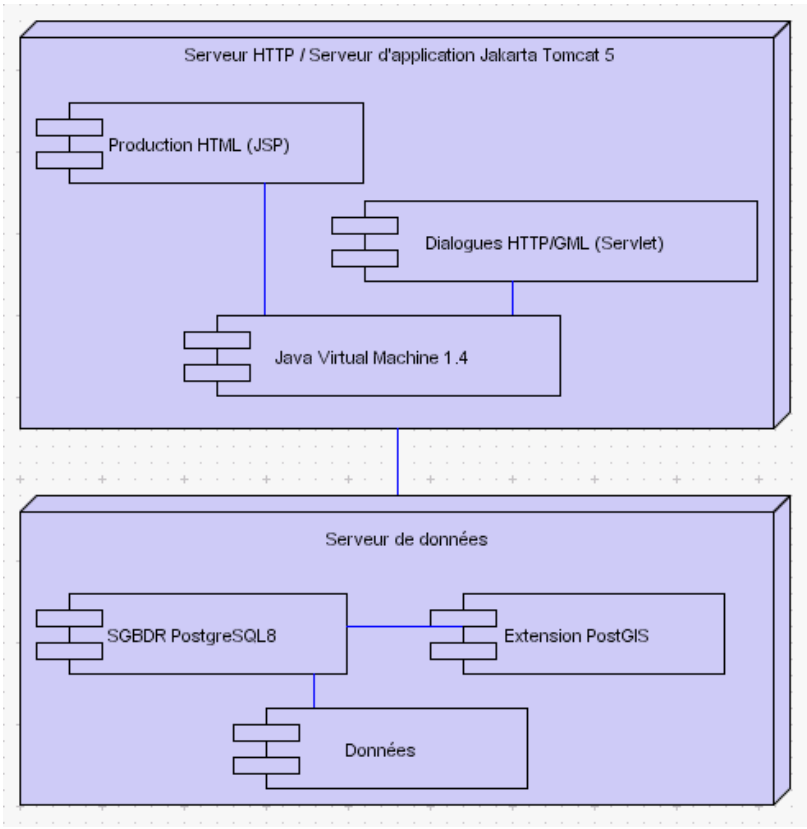
Modélisation du système 3
Partie Serveur 3
Partie Client 4
Description des sous composants 4
Production HTML (JSP) 4
Dialogues HTTP/GML (Servlet) 4
Navigateur Web 5
Applet Java 5
Interactions entre les sous composants 6
Accueil d'un internaute/identification d'un utilisateur 6
Visualisation/annotation d'une carte et enregistrement des modifications 7
Visualisation d'une carte 9
Enregistrement de nouvelles annotations 10
Description des données dans les documents GML 10
Niveaux d'accès aux pages et fonctionnalités 11
Partie Serveur/Base de données 12
Schéma de l'interaction 12
Phase de Tests 16
Planification de la phase de test 16
Ressources 17
Conclusion 18

Modélisation du système

Partie Serveur

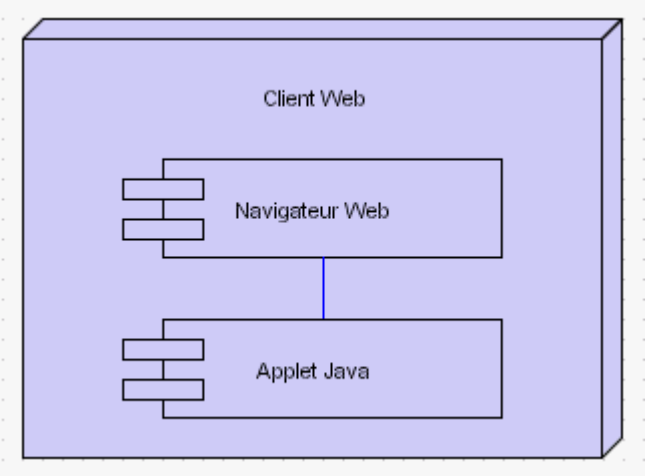
La partie Serveur regroupe 2 principaux sous composants interconnectés :

1. Un serveur HTTP / Serveur d'application :
    a. Des composants de **Production HTML** codés en JSP permettant l'interaction entre l'internaute et notre système.
    b. Des composants pour le **Dialogue http/GML** sous forme de Servlet Java dont les principales fonctions sont celles d'assurer la communication entre l'internaute et le serveur.
    c. La machine virtuelle Java **JVM 1.4** sur la quelle sont basées les deux composants précédents.
2. Un serveur de données :
    a. Un système de gestion de données relationnelle **PostgreSQL8**.
    b. Une extension géographique **PostGIS**.
    c. Les **données** géographiques elles-mêmes.



Partie Client

La partie client est constitué d'un navigateur Internet (MS Explorer ou Mozilla) supportant les applets Java. En effet, même si l'applet Java est téléchargée depuis le serveur, celle-ci fonctionnera du coté du client et dialoguera avec le serveur http.



Description des sous composants

Production HTML (JSP)

Description : Ensemble de composants permettant l'exploitation des fonctionnalités offertes à l'internaute : Identification, Recherche, Affichage de la liste des cartes annotées, …

En entrée : ils reçoivent le nom d'un service (fichier *JSP*) ainsi que des paramètres.
En sortie : ils génèrent du *HTML* lisible par le browser internet.

Ces composants utiliseront des *Beans* chargés d'interroger la base de données.
Les réponses aux requêtes doivent être fournies dans des délais courts (t < 3sec)

Dialogues HTTP/GML (Servlet)

Description : Ensemble de composants chargé d'assurer la communication entre le client et le serveur.

- Obtenir le document *GML* associé à une zone géographique

En entrée : Une requête de recherche avec le « bounding-box » de la zone.
En sortie : Un document *XML*.

Accès à la base de données *PostgreSQL* par l'intermédiaire de *JDBC* et interrogation de la base en utilisant les extensions géographiques *PostGis*, la bibliothèque *JDom* est utilisée pour générer le document *XML*.

- Réception un document GML d'annotation d'une carte.

En entrée : Un document *XML*.
En sortie : Un message nous indiquant si la mise à jour s'est bien effectuée ou non.

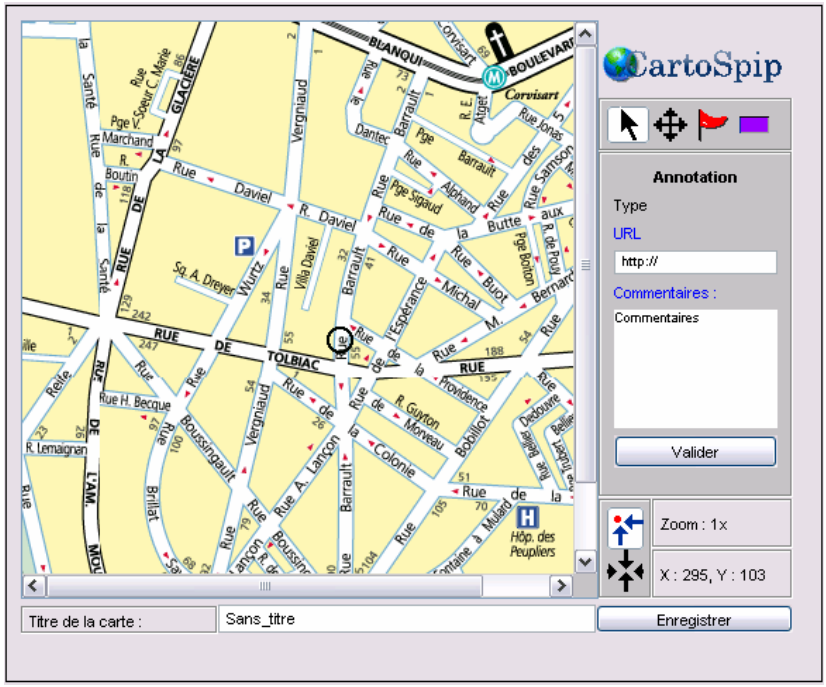
Le document *XML* est « parser » en utilisant *JDom*. Ensuite la base de données est mise à jour (*JDBC*) en fonction du contenu du document *XML*.

Navigateur Web

Description : Browser Internet permettant l'accès aux services proposés par CartoSpip.

Applet Java

Description : Extension du navigateur permettant la visualisation et l'annotation de cartes.

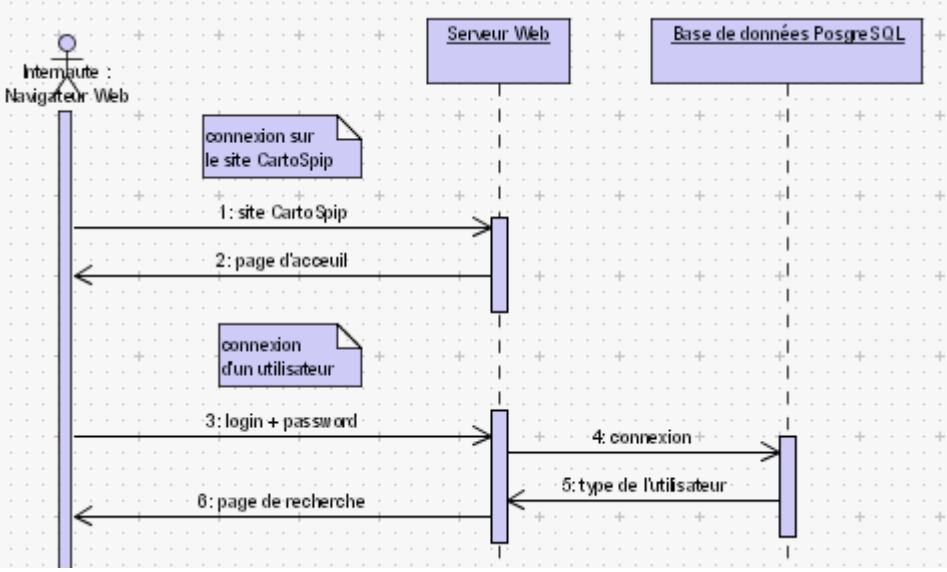


En entrée : L'applet est lancée avec en paramètre une requête correspondant à la recherche. L'applet télécharge une image JPEG/PNG depuis le serveur en utilisant cette requête et également un document XML (standard GML) qui contient les différentes annotations (drapeaux, zones, url et commentaires).

En sortie : Une fois que l'internaute aura ajouté ces annotations, celui-ci donnera un nom à la carte et déclenchera l'enregistrement de cette dernière. L'applet enverra un document XML (contenant les dernières annotations de l'internaute) au serveur. Un message de confirmation indiquera à l'utilisateur si l'enregistrement s'est bien effectué ou non.

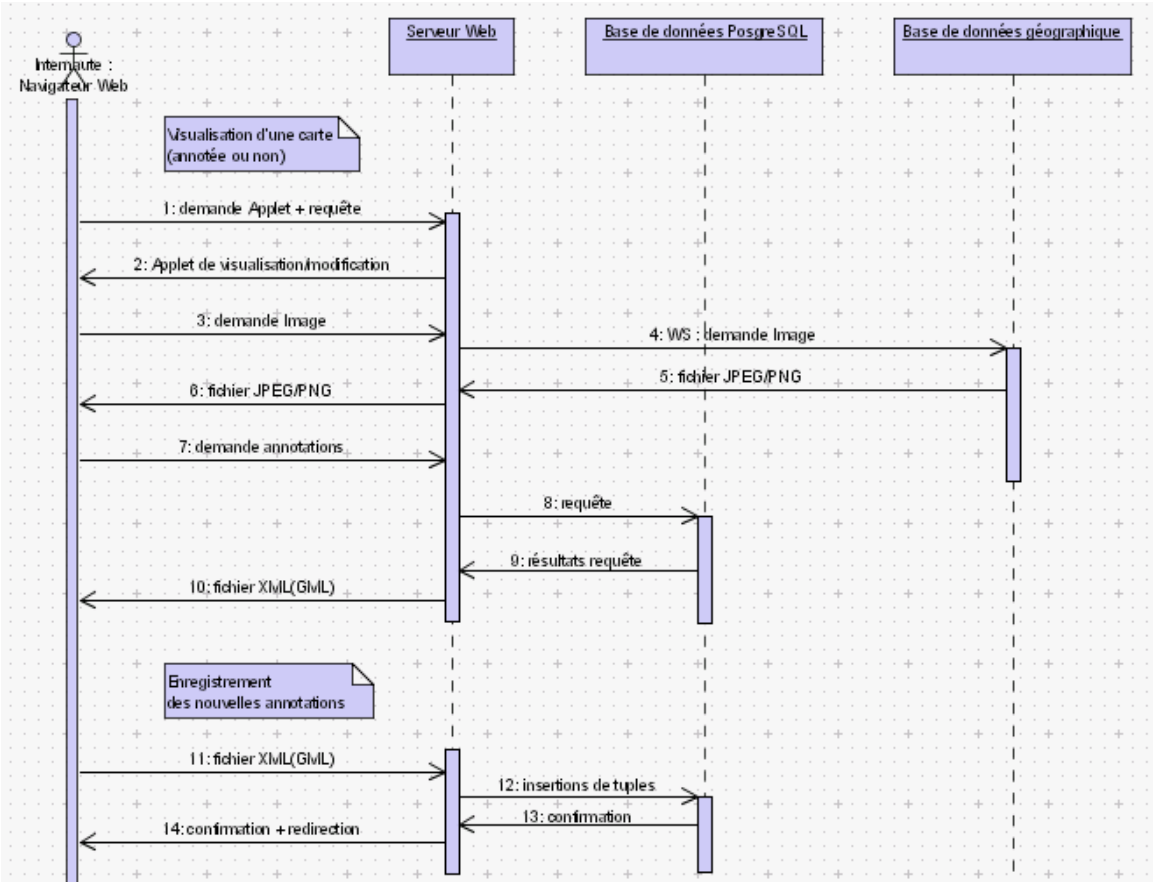
Interactions entre les sous composants

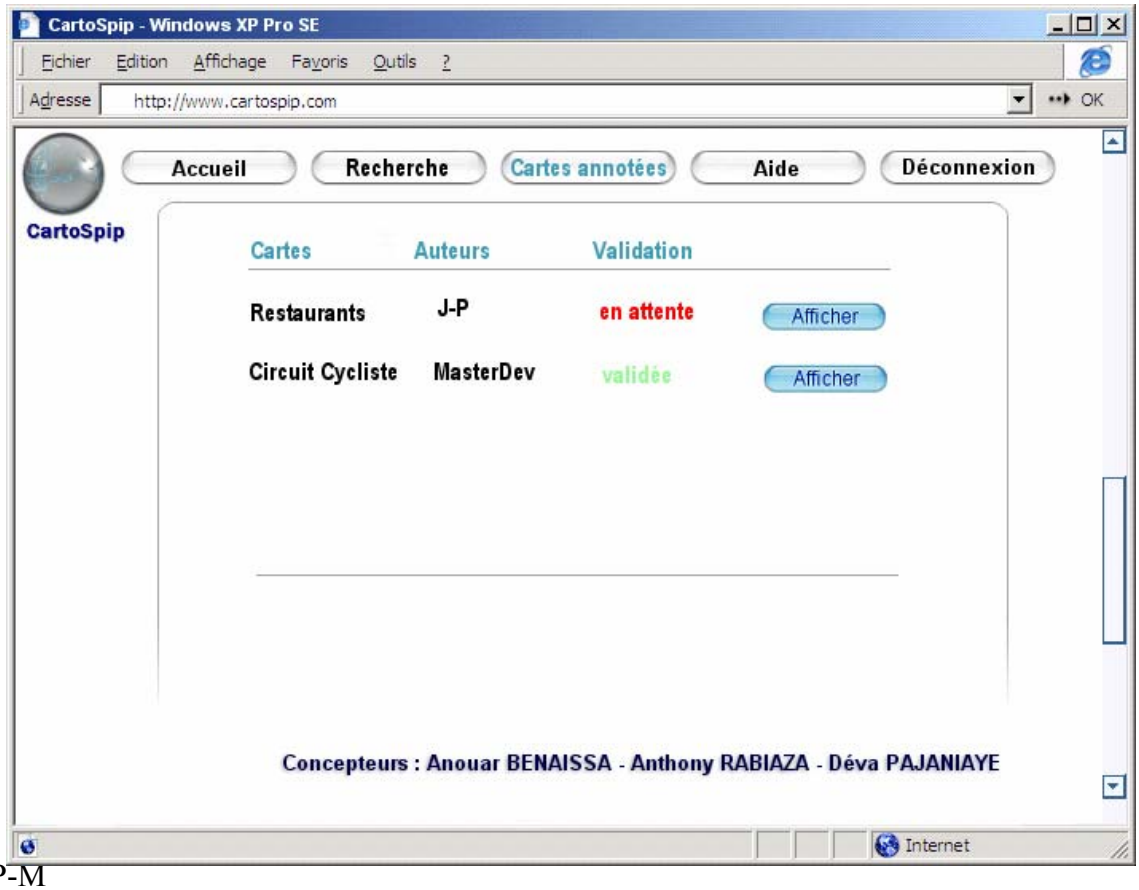
Accueil d'un internaute/identification d'un utilisateur

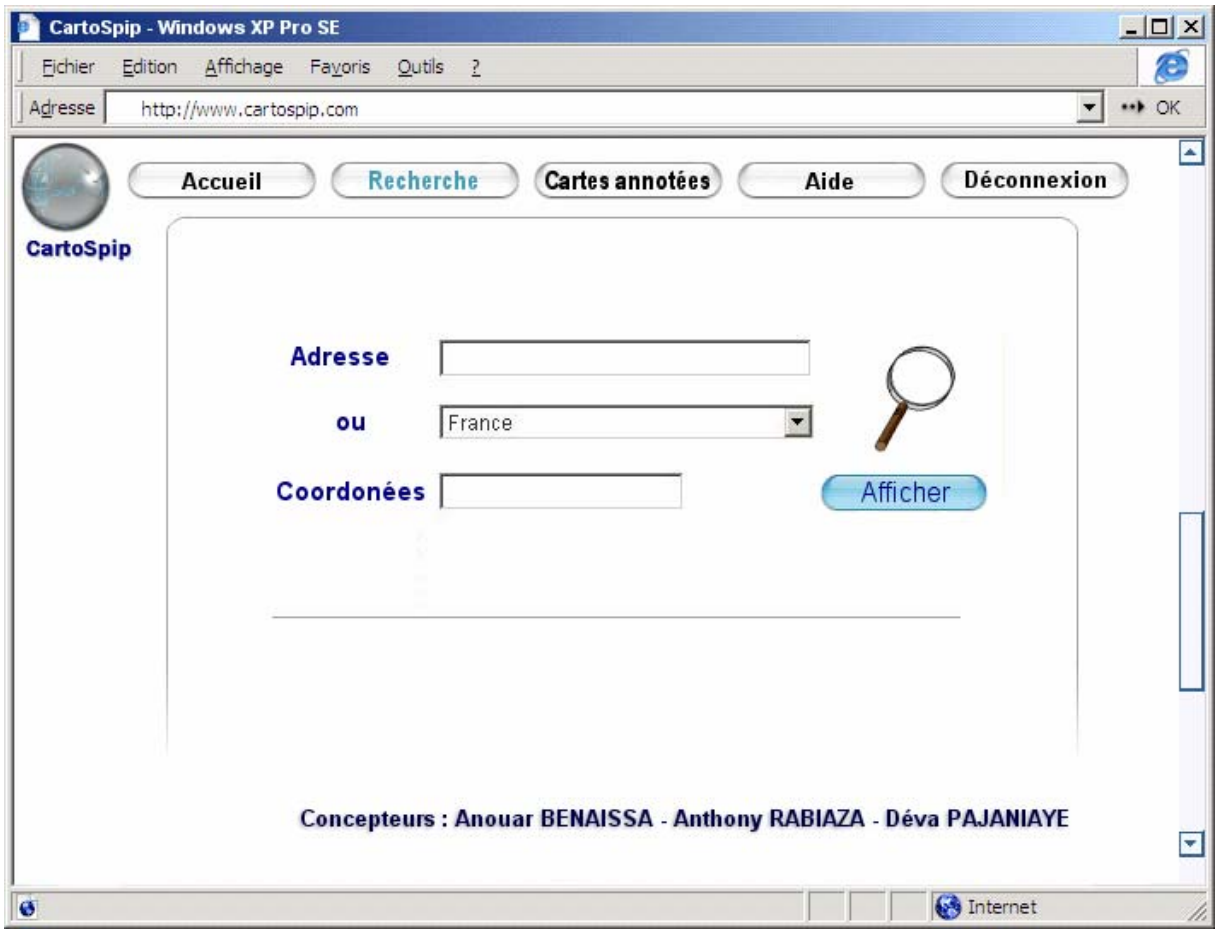




Visualisation/annotation d'une carte et enregistrement des modifications



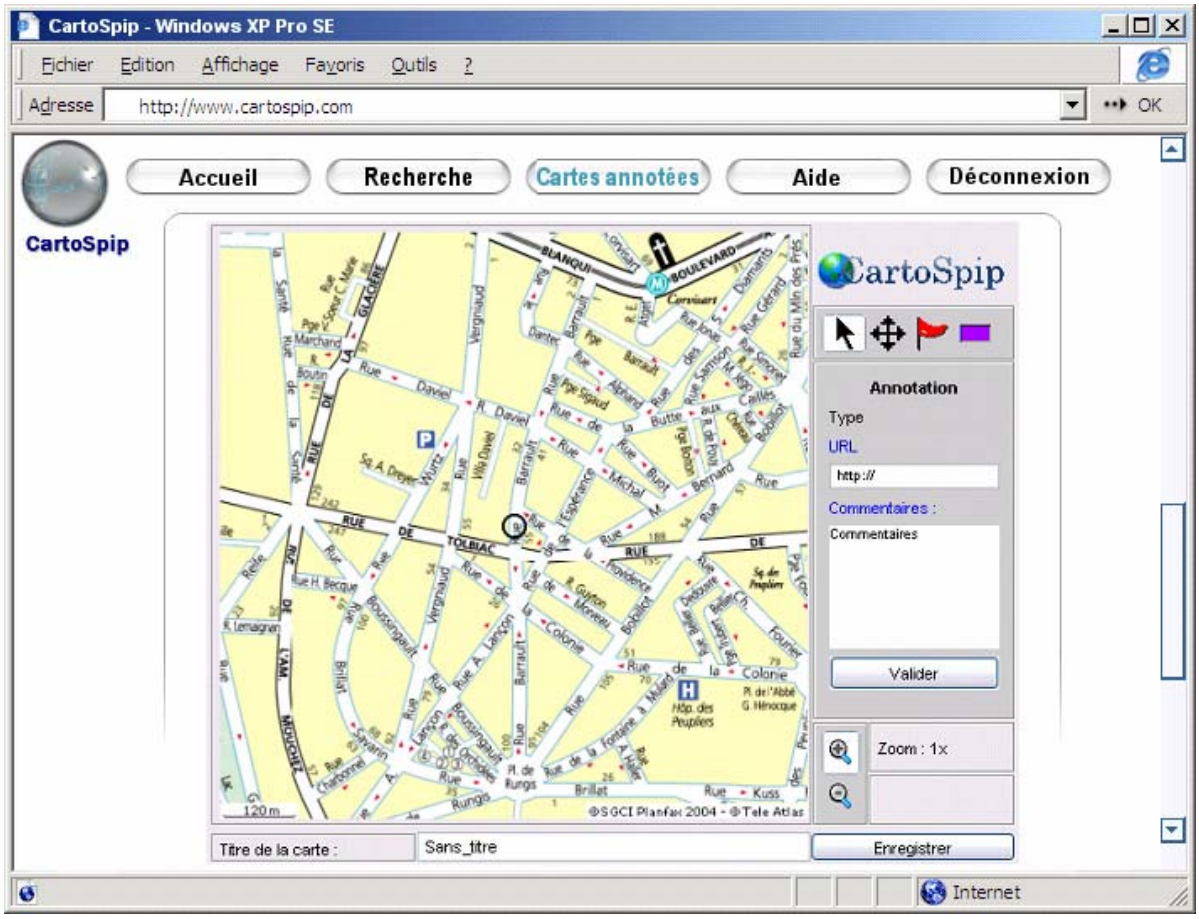



CartoSpip - Windows XP Pro SE
Fichier Edition Affichage Favoris Outils ?
Adresse http://www.cartospip.com
OK
Accueil
Recherche
Cartes annotées
Aide
Déconnexion
CartoSpip
Adresse
ou
France
Coordonées
Afficher
Concepteurs : Anouar BENAISSA - Anthony RABIAZA - Déva PAJANIAYE
Internet

Visualisation d'une carte

L'internaute a effectué une recherche ou demande à visualiser une carte annotée.

- Le navigateur envoie une requête http au Serveur Web pour obtenir l'image correspondant à une recherche.
- Le Serveur Web réceptionne la demande et transmet la requête à un service dédié (une base de données géographiques accessible par Web Services), celui-ci génère le fichier Image correspondant à la requête et le retourne au serveur web.
- Le navigateur récupère l'image présente sur le serveur.
- Une fois, l'image réceptionnée, le navigateur envoie une nouvelle requête au Serveur concernant les annotations de la carte.
- Le Serveur Web (un servlet spécialisé) réceptionne la demande et accède à la base de données géographique. Il effectuera des requêtes SQL pour récupérer les données correspondantes à la recherche et construit un document XML (à l'aide de JDom) répondant au standard GML.
- Le navigateur récupère le document XML et le « parse » afin d'afficher les différents objets géo référencés « au dessus » de l'image représentant la carte.



Enregistrement de nouvelles annotations

L'internaute annote une carte (vierge ou déjà annotée) et demande un enregistrement.

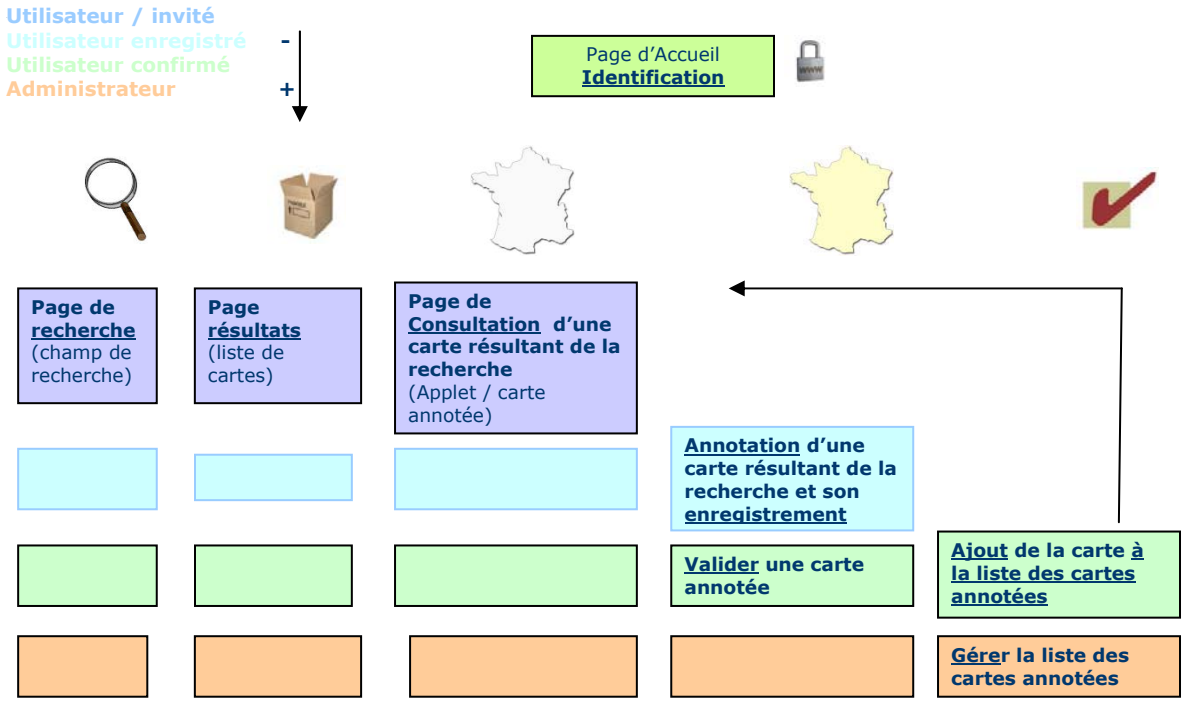
- A chaque ajout d'un objet graphique (drapeaux, …), l'applet stocke les différentes méta données relatives à cet objet.
- A la demande d'enregistrement, l'applet construit à l'aide de JDom un document XML contenant les ajouts de l'internaute et uniquement ces derniers. En effet, pour une carte non vierge, le GML ne contiendra pas les annotations faîtes par les autres internautes.
- Une fois ce document construit il l'envoie au serveur web.
- Le serveur Web par l'intermédiaire d'un servlet, lit le flux binaire et récupère le document XML correspondant. Le servlet va ensuite « parser » ce document et accéder à la base de données pour y effectuer les insertions correspondantes.

Description des données dans les documents GML

Les documents GML échangés entre le client et le serveur web contiennent les données suivantes :

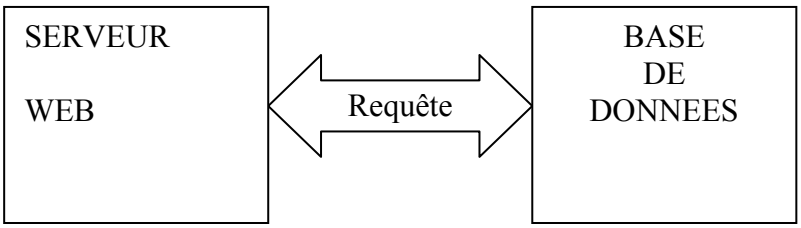
- Référence
- Coordonnées Objets par rapport à la référence
- Méta données de l'objet (URL, commentaires, …)
- Méta données sur l'ajout (nom utilisateur, nom de la carte annotée, date, …)

Niveaux d'accès aux pages et fonctionnalités

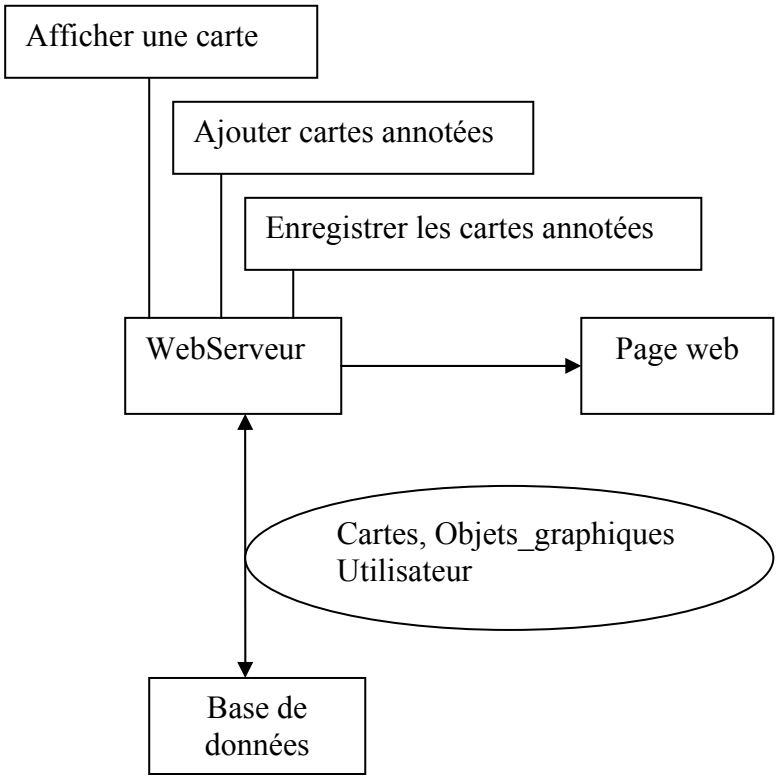


Partie Serveur/Base de données

Schéma de l'interaction



Requêtes : Sélectionner, enregistrer ou supprimer des lignes de tables



Méthodes : méthodes utilisant la base de données et Web
Identifier
Insérer des cartes annotées,
Mise en attente de validation de cartes annotées,
Afficher Modifier, Supprimer des cartes annotées

Pour construire les tables, on se base sur les fonctionnalités que doit faire le site :

=>Connexion
=>Afficher/lister, enregistrer, supprimer et modifier les cartes annotées non validées ou validées

1- Connexion
Pour se connecter, l'utilisateur aura besoin de son login et de son mot de passe.
La méthode retournera le type d'utilisateur.
Pour ce faire, on en déduit la table suivante
Utilisateur (**login**, pass, type, nom, prénom, titre, Tél)

On aura 3 types d'utilisateurs :

- Le superUser qui a tous les droits (enregistrer, supprimer)
- L'utilisateur enregistré
- L'invité

2- Afficher/lister, enregistrer et modifier les cartes annotées

Pour utiliser ces méthodes, on aura besoin des tables suivantes :

- Carte (**Num,** nom, adresse)
- Objet (**Nom**, couleur)
- Coordonnée (**X, Y**)
- Référence **(Nom**, coordonnée)
- Objet_Graphique (**objet, coordonnée, référence,** commentaires)
- Carte_A(**carte, objet_Graphique, concepteur**, état, commentaires)

Concepteur : l'utilisateur qui a modifié ou supprimé sa carte annotée, il est le seul à pouvoir le faire.
Ce qui peut être envisagé est qu'un utilisateur non concepteur de la carte pourra faire une copie d'une carte annotée qui n'est pas le sien pour éventuellement le modifier ou supprimer. (Possible car sera le sien)

État : Le champ état dans la table Carte_A peut prendre 2 valeurs : valide ou non valide.
L'utilisateur confirmé est un utilisateur de type « valideur ». Il pourrait être un chef hiérarchique ou un moniteur. Il aura la possibilité de voir, de valider s'il le veut les cartes annotées.

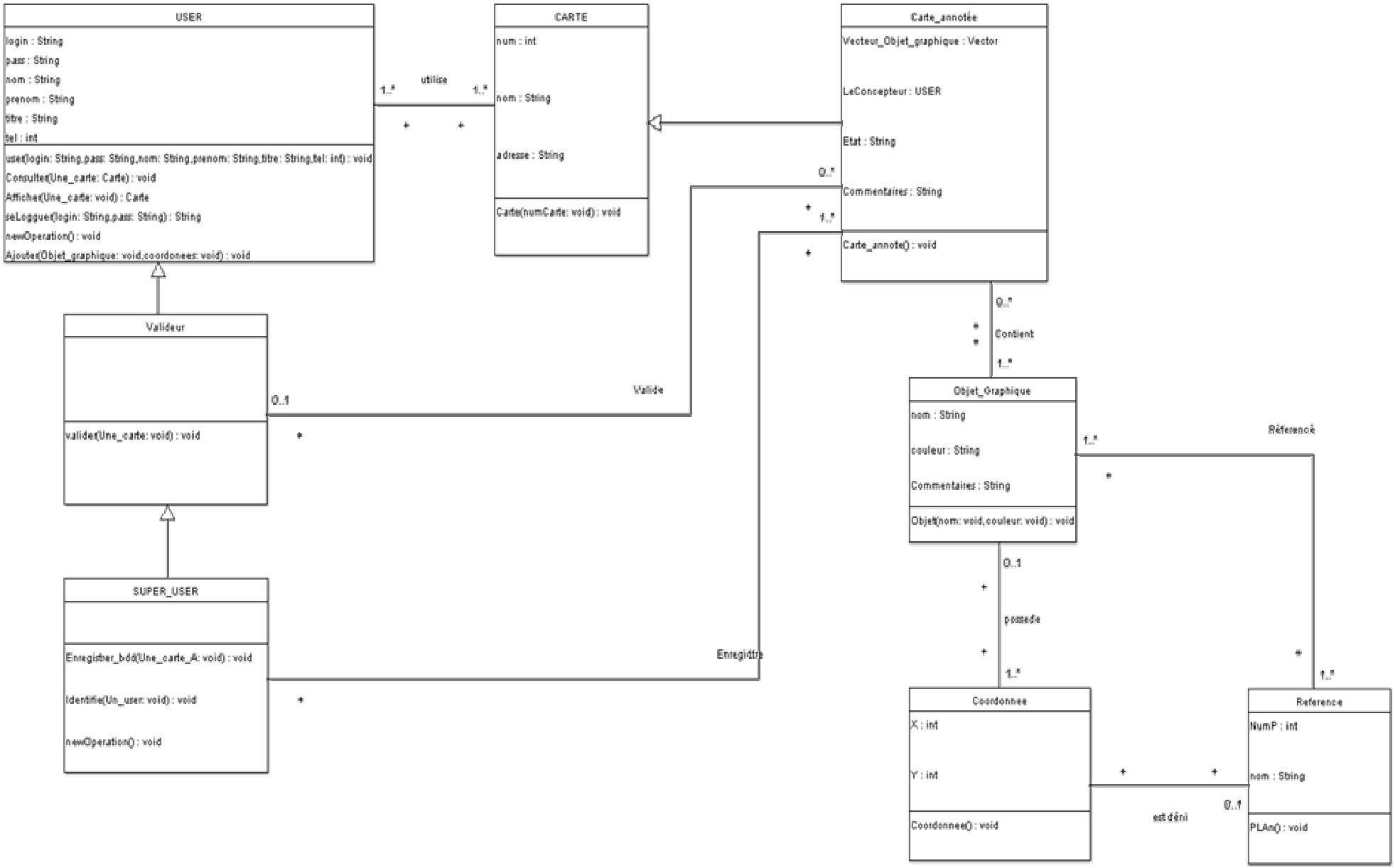
Une référence correspond à l'origine d'un plan de coordonnées (Xo, Yo)

Le champ commentaires dans les tables Carte_A et Objet_Graphiques permettra de stocker le commentaire de l'utilisateur concevant la carte, au niveau de chaque objet et de la carte elle-même.
Elle pourra être visualisée quand le curseur pointera l'objet.

Un objet graphique peut avoir plusieurs coordonnées pour une référence donnée et peut avoir plusieurs références.

Pour des raisons de surcharge de tables, un utilisateur pourra modifier **N** cartes donc N cartes annotées.
Le paramètre N sera fixé par le client.

Dans la page suivant nous avons le schéma entité/association fait avec ArgoUML :


USER
login : String
pass : String
nom : String
prenom : String
titre : String
tel : int
user(login: String,pass: String,nom: String,prenom: String,titre: String,tel: int) : void
Consulter(Une_carte: Carte) : void
Afficher(Une_carte: void) : Carte
seLoggue(login: String,pass: String) : String
newOperation() : void
Ajouter(Objet_graphique: void,coordonnees: void) : void
utilise
1..*
1..*
CARTE
num : int
nom : String
adresse : String
Carte(numCarte: void) : void
Carte_annotée
Vecteur_Objet_graphique : Vector
LeConcepteur : USER
Etat : String
Commentaires : String
Carte_annote() : void
0..*
1..*
Valideur
valide(Une_carte: void) : void
0..1
Valide
0..*
Contient
1..*
Objet_Graphique
nom : String
couleur : String
Commentaires : String
Objet(nom: void,couleur: void) : void
Référencé
1..*
SUPER_USER
Enregistrer_bdd(Une_carte_A: void) : void
Identifie(Un_user: void) : void
newOperation() : void
Enregistre
0..1
possede
1..*
Coordonnee
X : int
Y : int
Coordonnee() : void
est défini
0..1
Reference
NumP : int
nom : String
PLAn() : void
1..*

Phase de Tests

- JSP
  - Identification
  - Recherche
  - Liste cartes annotées

- Servlet (Http/GML)
  - Récupération d'images suite à une requête
  - Récupération du GML d'une carte annotée
  - Enregistrement des modifications sur une carte
  - Envois du GML

- Interaction avec la base de données
  - Serveur/Client et Serveur/Bases de données

Planification de la phase de test

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

Ressources

- **Serveur Jakarta-Tomcat 5.0** : Serveur HTTP et serveur d'applications Java, celui-ci comporte :
    - Les JSP pour la publication HTML.
    - Les Servlets utilisés pour les dialogues avec le client. Les API **JDom** et **Common FileUpload** seront utilisés dans ces Servlets, ces derniers servent respectivement à la génération/lecture des flux XML et à la réception de flux http.

  Ces derniers sont basés sur **Java SDK 1.4.2** : l'environnement de développement multi-plateformes de Sun.

- Le système de gestion de base de données relationnelles **PostGreSQL 8.0** et son extension géographique **PostGIS 0.9**.

  Les données utilisateurs et géographiques seront stockées sur cette base et interrogeables en SQL.

  L'interconnexion entre le serveur principal et le serveur de données se fait par l'intermédiaire des API **JDBC** écrites en Java.

- Le navigateur de type **MS Explorer** (5 ou 6) ou **Mozilla** (1.6) communiquera en http avec le serveur et l'applet Java exploitera la bibliothèque **Swing** pour l'affichage de composants graphiques et de la bibliothèque **JDom** pour la gestion des flux XML (GML).

Conclusion

Suite à la phase planification, nous avons présenté ici le volet analyse de notre projet qui à traité de la modélisation du système et de la description de ses sous composants en focalisant sur les Interactions entre ces derniers.

Cette phase analyse précède la phase de codage qui va s'étaller de mi-novemebre à mi-décembre. Ceci nous a permis de structuré la conception et l'implémentation de notre système.

La phase de tests quant à elle est programmée pour début à mi-décembre, et elle couvrira aussi bien la partie Serveur/Client que celle du Serveur/Base de données.