Rapport du TER FMIN200
du Master IFPRU
du 16 Janvier au 25 Avril 2008

Représentation et visualisation des résultats de recherche Web sur carte géographique Google Maps

Encadrant: Jean-Yves DELORT

Etudiants: Abdoulaye DIANKHA
Adel ABDELAZIZ
Alexis CRETINOIR
Saber CHIHOUB

Remerciements

Nous tenons à remercier notre responsable de projet Mr Jean-Yves Delort pour son aide, ses directives et pour l'implication qu'il a manifesté tout au long de ce projet.
Merci aussi aux autres professeurs qui nous ont permis, grâce à leurs cours, de réaliser ce TER, à Mr Pierre Pompidor pour les cours de Galaxie XML et Mme Thérèse Libourel pour les cours de base de données spatial.

Enfin nous remercions l'Université Montpellier 2 de nous proposer de présenter un projet de Travail d'Etude et de Recherche, ce qui nous permet de nous spécialiser dans un domaine et de l'explorer en détail.

Sommaire

1.INTRODUCTION

Les Services Web sont une technologie permettant à des applications de dialoguer à distance via Internet et ceci indépendamment des plates-formes et des langages sur lesquelles elles reposent. Pour ce faire, les Services Web s'appuient sur un ensemble de protocoles standardisant les modes d'invocation mutuels de composants applicatifs.

De plus en plus, nous trouvons des Services Web orientés vers les Systèmes d'Information Géographique. Aujourd'hui, l'évolution des SIG tend vers une accessibilité pour le web avec les Web Mapping, en particulier avec Google Maps.

L'évolution technologique des SIG met désormais à disposition des utilisateurs des outils en ligne (internet/intranet). La mise en œuvre de telles solutions réclame un panel de compétences variées (SIG, internet, développement, sémiologie graphique, communication...).

La puissance actuelle des Services Web réside dans les applications composites, en combinant des éléments provenant de sources multiples derrière une interface graphique unifiée.

Le travail à effectuer dans ce TER est un travail exploratoire qui consiste à construire un Mashup sur l'API Google Maps. En d'autres termes, il s'agit d'une appropriation de Google Maps, afin de disposer de ses fonctionnalités et de son service cartographique pour construire un géo-service personnalisé proposant des améliorations de navigation et de visualisation des résultats de carte géographique et d'intégrer d'autre Services Web.

1.1 Le web 2.0

Les technologies liées au développement des interfaces clientes pour le web viennent de connaître une forte amélioration avec la survenue de qui est convenu d'appeler le Web 2.0. En effet, outre l'interactivité accrue des transactions opérées entre le clients et le serveur grâce à de nouvelles pratiques comme celle d'Ajax, ou la sophistication des interfaces couplée avec l'intégration des back-ends au niveau des serveurs de présentation, bien illustrées par la montée en puissance des codes Flash générés par les serveurs de présentation Flex ou Open Laszlo, la principale révolution provient de l'externalisation des applications web qui sortent du navigateur pour être exploitées par un client riche reposant sur un environnement d'exécution déjà présent sur le poste client.

1.1.1 Les Technologies offertes par le web 2.0

Nous assistons avec le Web 2.0 à deux axes de développement ayant pour but de dépasser les frustrations qui sont dans le Web 1.0. La mise en place d'une interactivité efficiente entre le client, le serveur et la génération d'interface aussi sophistiquées que celles qui pouvaient l'être avec les clients lourds, avec les applications web riches RIA (Rich Internet Application) et Les Applications de bureau riches RDA (Rich Desktop Application).

1.1.2 Les applications web riches

Les Applications Web Riches (RIA) sont des ensembles d'applications très riches qui rassemblent plusieurs langages parmi lesquels nous pouvons citer :

- **AJAX** (Asynchronous JavaScript and XML, 2005), le langage JavaScript acquière une nouvelle dimension en permettant au client, via l'implémentation de l'objet XMLhttpRequest d'effectuer la mise à jour et la validation, en temps réel des données intégrées dans la page web. Cette technologie fortement à la mode est portée par l'implication de Google par le biais d'applications phare (Google Maps, Google mail, Google Earth).
- **FLASH** (Adobe Flash anciennement Macromedia Flash 1996), était à l'origine un environnement de développement intégré (IDE) permettant de générer des animations, et plus généralement des objets graphiques interactifs insérables dans des pages web. Mais la révolution vient du fait que le pseudo-code Flash est maintenant généré à partir d'une spécification XML exploitée par un serveur de présentation.
- **FLEX** fait appel au langage MXML (Macromedia XML, basé sur XML) et à Action Script 3.0 pour la gestion de l'interactivité, tandis qu'Open Laszlo utilise sa propre spécification XML et JavaScript. Ces deux serveurs permettent la gestion du cache et l'intégration des back-end.
- **XUL** (XML-based User interface Langage, 2001) une réalisation du projet mozilla ayant pour but de promouvoir des technologies clientes ouvertes et conforme aux standards c'est-à-dire intégrables. Le langage XUL permet de définir tous les composants graphiques qui peuvent se rencontrer dans une page : zone d'édition, boutons, listes menus ….
- **SVG** est une application du W3C également basé sur XML, elle permet de décrire des ensembles de graphismes vectoriels.

Ces deux technologies sont extrêmement importantes car elles ont initié la spécification ouverte et interopérable des interfaces spécifiés par XML.

1.1.3 Les Applications de bureau riches

Les applications de bureau riches (RDA) s'affranchissent du navigateur pour être gérées par d'autres clients. C'est aussi que ces applications externalisées du navigateur sont également dénommées clients riches. Ces applications reposent sur l'environnement d'exécution du Framework java pour Eclipse et JavaFX, du framework.NET pour SilverLigth et de la nouvelle plateforme adobe pour AIR.

- **JavaFX** (mai 2007) s'appuie sur java FX script un langage de script a mi-chemin entre JavaScript et java, et s'appuyant sur les bibliothèques java2D Swing.

- **SilverLigth 1.1** s'appuie sur l'environnement d'exécution (CLR : Common Language Runtime) du Framework.NET, et permet le développement d'applications riches de bureau. Le développement des projets SilverLigth est préconisé sous Visual Studio 8.

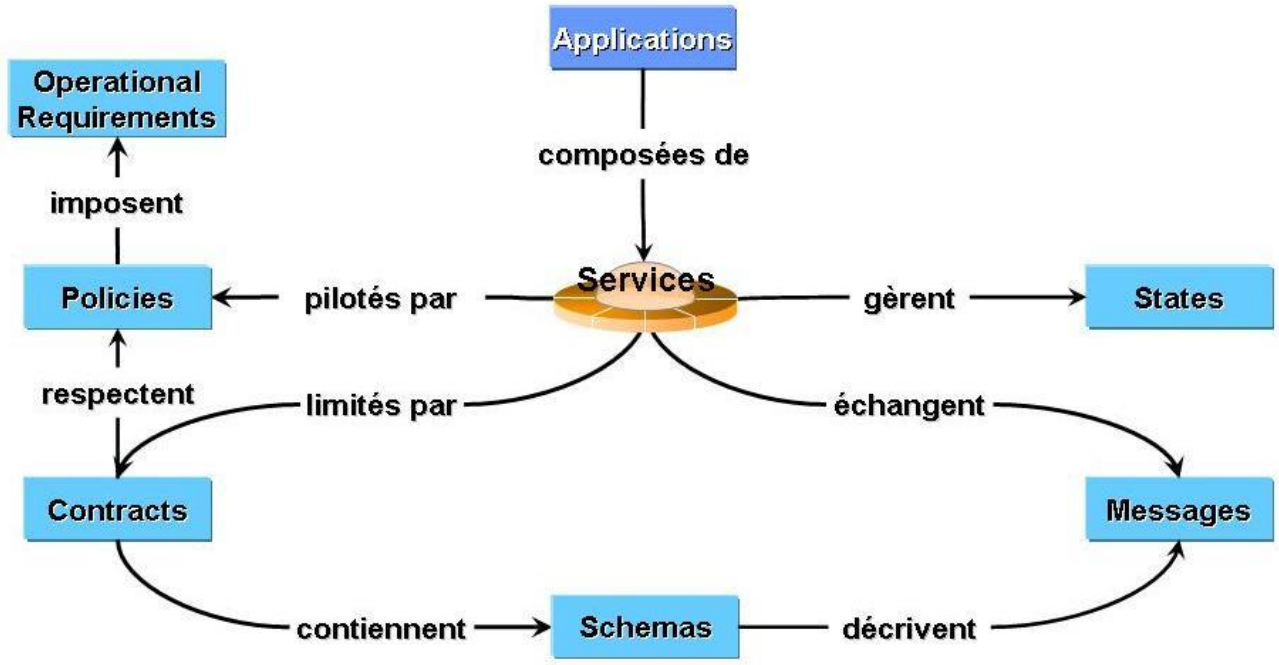
- **AIR Web 2.0** (adobe integrated Runtime, ex Appolo juin 2007) est une plateforme qui gère les technologies HTML/CSS, Ajax, et évidement Flash/Flex pour déployer des applications internet riches sur les postes de travail en intégrant une base de données locale.

1.2 Le Service Web

Le Service Web est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur Internet ou sur Intranet, par et pour des applications ou machines, et en temps réel pour les besoins des utilisateurs. Il existe plusieurs technologies derrière le terme Services Web:

- Les Services Web de type REST exposent entièrement ces fonctionnalités comme un ensemble de ressources (URI) identifiables et accessibles par la syntaxe et la sémantique du protocole HTTP. Les Services Web de type REST sont donc basés sur l'architecture du Web et ses standards de base sont: HTTP et URI.
- Les Services Web WS-* exposent ces mêmes fonctionnalités sous la forme de services exécutables à distance. Leurs spécifications reposent sur les standards SOAP et WSDL pour transformer les problématiques d'intégration héritées du monde Middleware en objectif d'interopérabilité. Les standards WS-* sont souvent décriés comme l'étaient des technologies complexes héritées du vieux principe RPC, fortement couplées et difficilement interopérables dans des environnements hétérogènes. L'OASIS et le World Wide Web Consortium (W3C) sont les comités de coordination responsables de l'architecture et de la standardisation des Services Web. Ces Services Web WS-* sont par ailleurs définis selon le type d'architecture SOA.

1.2.1 Le concept de SOA



Le service est l'unité atomique d'une architecture SOA. Une application est un ensemble de services qui dialoguent entre eux par des messages. Le service est un composant clef de

l'Architecture Orientée Services. Il consiste en une fonction ou fonctionnalité bien définie. C'est aussi un composant autonome qui ne dépend d'aucun contexte ou service externe.

Un service est une entité de traitement qui respecte les caractéristiques suivantes :

Large Granularité (coarse-grained) : Les opérations proposées par un service encapsulent plusieurs fonctions et opèrent sur un périmètre de données large au contraire de la notion de composant technique.

Interface : Un service peut implémenter plusieurs interfaces, et aussi plusieurs services peuvent implémenter une interface commune.

Instance unique : A la différence des composants qui sont instanciés à la demande et peuvent avoir plusieurs instances en même temps, un service est unique. Il correspond au design pattern Singleton.

1.2.3Les protocoles et les normes de SOA

Les architectures SOA reposent principalement sur l'utilisation d'interface d'invocation (SOAP, Simple Object Access Protocol) et de vocabulaire de description de données (WSDL, Web Services Description Language et XML, eXtensible Markup Language) qui doivent être communs à l'ensemble des agents (fournisseurs de services et utilisateurs de services).
Ce dispositif permet de réutiliser les applicatifs métiers, le but étant de permettre à l'entreprise de s'adapter rapidement à un nouveau contexte de marché. Parmi les différentes couches de normes et protocoles qui permettent de bâtir de telles architectures, on relève :

- la gestion d'un annuaire de services (quels sont les services mis à disposition et par qui) avec : UDDI (Universal Description Discovery and Integration) normalisé par l'OASIS.
- la description des interfaces des services (quelles sont les données nécessaires à l'exécution du service) avec : WSDL recommandé par le W3C.
- l'invocation (ou l'appel) du service (la requête transmise au service) avec : SOAP recommandé par le W3C.
- le format des données échangées avec : XML recommandé par le W3C.
- le transport des données avec les protocoles internet : HTTP et TCP/IP qui sont des normes RFC.
- la gestion de la sécurité avec : SSL (Secure Sockets Layer), XML Signature, XML Encryptions, SAML (Security Assertion Markup Language) ou encore XKMS (XML Key Management Spécification, qui gère les infrastructures à clé publique ou PKI).
- la gestion transactionnelle (gestion du protocole de validation à deux phases, two-phase commit, pour la mise à jour contrôlée de plusieurs bases de données réparties entre plusieurs fournisseurs de services.

1.3 Le Géo-Service

Le Géo-Service est un ensemble de méthodes et de services permettant à toute personne de pouvoir rechercher, afficher, manipuler, visualiser, distribuer, d'envisager les échanges et le partage de données géographiques .

D'après la loi sur la Geoinformation (LGeo) qui est entrée en vigueur en janvier 2008, autorisant à toutes les entreprises et sociétés qui travaillent sur les données geographiques de le mettre à la disposition de tout un chacun pour une vision des échanges et la logique de point d'entrée unique pour accéder aux geodonnees du territoire via un geoportail, en insistant sur le caractère interopérable et normalisé de ces services.

Parmi les principaux Géo-Services de base nous pouvons citer :

- Le service de métadonnées.
- Le service de catalogage (Géo-Catalogue).
- Le service de Web-Mapping.
- Le service de distribution (Géo-Portail).
- Le service de transformation (Géo-Commande).

Un Système d'Information Géographique (SIG) est l'ensemble des matériels, des logiciels, des données, des personnes, et des compétences mises en place pour analyser un territoire. L'offre en matière de SIG libre est assez vaste et bien qu'elle souffre de quelques défauts, elle continue d'évoluer.

Deux modes de représentations sont possibles :

- **Vectoriel** (format vecteur) : les objets sont représentés par des points, des lignes, des polygones ou des polygones à trous.
- **Matriciel** (format raster) : il s'agit d'une image, d'un plan ou d'une photo numérisés et affichés dans le SIG en tant qu'image.

1.4 Normes OGC

Open Geospatial Consortium (OGC) est une organisation internationale à but non lucratif dédiée au développement des systèmes ouverts en géomatique. Cet organisation vise à fixer des normes et des standards pour le contenu et les services géospatiaux, les SIG et les échanges informatiques.

Les principaux buts de cette organisation sont:

- De promouvoir l'utilisation d'application ouvertes.
- De synchroniser les technologies de l'information géographique avec des standards.
- D'assurer la coopération entre les fournisseurs et les utilisateurs.
- D'impliquer l'ensemble de la communauté dans le processus d'interopérabilité.
- De fournir une plate-forme d'échange pour promouvoir les développements communs.

Les recommandations les plus importantes faites par l'OGC sont:

- Web Map Service (WMS).
- Web Feature Service (WFS).
- Web Coverage Service (WCS).
- Catalog Service Web (CS-W).
- Simple Features - SQL (SFS).
- Geography Markup Language (GML).
- Sensor Web Enablement (SWE).

2.ARCHITECTURE GOOGLE MAPS

Google Maps utilise le serveur cartographique Drill Down Server de la start-up Telcontar (maintenant DeCarta), également adoptée par Yahoo! et Ask Jeeves pour leur service respectif.

La plateforme est la même, ce qui change, c'est l'interface graphique que ces organismes développent eux-mêmes pour leurs utilisateurs.

Pour atteindre un tel niveau de rapidité et d'interactivité, malgré la complexité de l'interface, les ingénieurs de Google utilisent le moteur XSLT d'Internet Explorer ou Firefox, pour convertir à la volée les données XML, transmises par les serveurs de Google en un mélange de codes JavaScript, CSS, HTML et DHTML compréhensibles par le navigateur.

2.1 Présentation de deCarta

DeCarta est la première plate-forme logicielle géospatiale sur laquelle s'appuient les applications de services de proximité (LBS) les plus répandues actuellement, notamment celles déployées par Google Maps, Yahoo, Ask.com, Verizon et Sprint. La technologie brevetée unique de la société est idéale pour les applications de services de proximité à volumes élevés, pour une utilisation dans des applications Internet, mobiles, de navigation personnelle et d'entreprise où l'extensibilité, la rapidité et la fiabilité sont vitales. Sa plate-forme logicielle géospatiale Drill Down Server, son moteur Rich Map Engine et son service Hosted Web Service sont privilégiés par les développeurs d'applications et les fournisseurs de services qui recherchent également la souplesse de personnalisation des styles de carte, des fonctions de création d'itinéraire uniques et de marque en nom propre.

2.1.1 Drill Down Server Web Server

Le DDS de deCarta est une plate-forme géographique s'appuyant sur un serveur et servant de base à de nombreuses applications géographiques parmi les plus répandues sur Internet et sur appareils mobiles. Il offre des performances très poussées et l'extensibilité pour les applications du marché grand public. Pour la plus grande joie des clients, il devient possible de pousser des services de proximité sur Internet vers le PND du client.

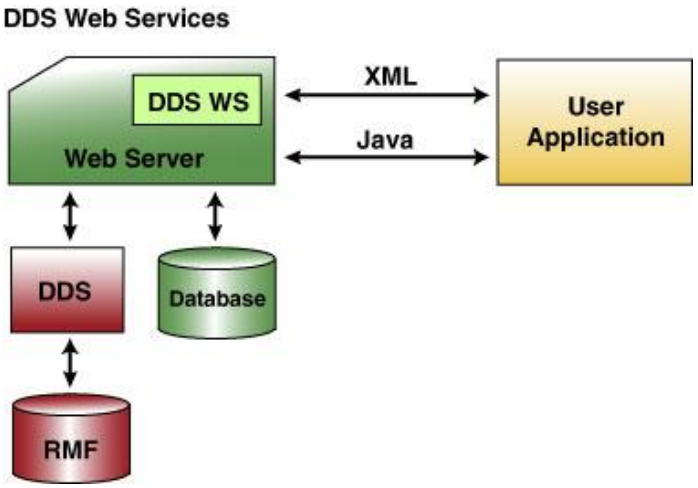


Figure 2.1: DDS Web Services

Pourquoi DDS est si rapide?

Au coeur de Decarta (TelContar) y a une nouvelle methode d'accés aux données spatiales qui et fondamentalement different de tout ce qu'on a essayer de faire jusqu'ici .
On a réorganisé les données provennont de toutes les sources dans un format appelé "Rich Map Format" (RMF) , nous avons 10 brevets acordés et autres 21 classés (déposés) dans ce secteur.

Une différence subtile, nous avons construit notre moteur et nos méthodes d'accés on été conçues pour faire des calcules d'itinéraire trés performants.

2.1.2 Quelques fonctionnalités de base

2.1.2.1 Le concept grille de dalles

La carte est composée d'une grille de m lignes et n colonnes, l'application choisit les valeurs de m et n. Cette grille est obtenue par la juxtaposition de m*n dalles chaque dalle est une image qui représente une portion de la carte, la largeur et la hauteur pour chaque image doit être la même. En plus des dalles qui se situent à l'intérieur de la grille, ils existent d'autres invisibles à l'extérieur qui sont chargées dans la mémoire, et qui représentent la continuité de la carte.

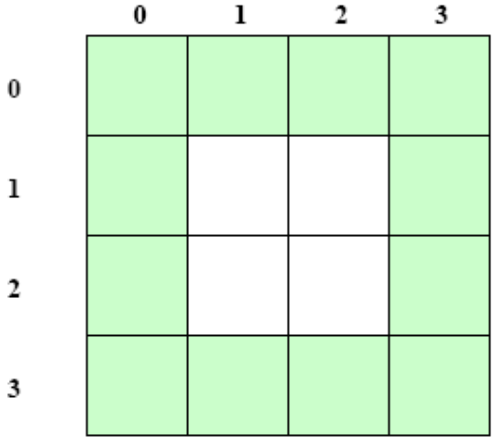


Fig. 1. A 4-row by 4-column grid of map tiles. The non-shaded tiles are "on screen". The remaining tiles are peripheral off-screen buffer. Row and column indexing are zero-based and originate in the north-west corner of the grid.

Lorsque l'utilisateur déplace la carte, ces dalles seront visibles, et les autres dalles seront chargées en mémoire. L'utilisateur aura l'impression que la carte existe sur son ordinateur, et n'attend pas un chargement.
Avec DDS Web Services, l'URL de chaque image peut être obtenue en utilisant une simple requête XML. Cette requête est illustrée ci-dessous listing 1.

```
1   <PortrayMapRequest>
2     <Output width="300" height="300" format="GIF">
3       <CenterAddress>
4         <Radius unit="KM">1</Radius>
5         <Address countryCode="US">
6             <freeFormAddress>
7                 4 n 2nd st, san jose ca
8             </freeFormAddress>
9         </Address>
10      </CenterAddress>
11      <TileGrid rows="4" columns="4"/>
12    </Output>
13  </PortrayMapRequest>
```

Listing. 1. A PortrayMapRequest with a TileGrid requested in the output. The server returns a PortrayMapResponse containing a TileGrid element populated with individual Tile elements.

2.1.2.2 La position spatiale de la grille des dalles

Contrairement au carte pré-rendu, DDS Web Services supportent le rendu immédiat des dalles. deCarta a opté pour sa nouvelle technologie CenterAdresse (Adresse Centrer), qui au lieu de faire deux requêtes au server:

- geocoder l'adresse en lat/lng.
- la lat/lng est utiliser pour parametrer «BoundingBox» ou bien «CenterContext».

Lance directement le seveur qui geocode automatiquement l'adresse et retourne une seule image ou une grill d'images qui est centrée sur le candidat d'adresse de qualité la plus haute. Cela permet à l'application de facilement présenter des choix d'adresse alternés à l'utilisateur.
Le CenterAddress et beaucoup d'autres fonctions du DDS Web Serveur ont été développées pour réduire le nombre de communications client-serveur et réduisent au minimum le code de traitement du client.

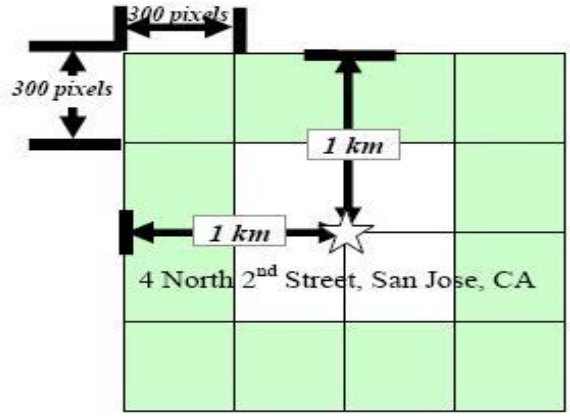


Fig. 2 – Pixel and spatial dimensions of a TileGrid centered at an address.

2.1.2.3 Paramètres Nord et Est

En réponse à une PortrayMapRequest, le serveur rend une PortrayMapResponse contenant un élément TileGrid rempli avec des éléments de dalles individuels. Plusieurs pièces d'informations importantes sont rendues dans le réseau de dalle :

1. La mesure spatiale complète du TileGrid.
2. La ligne et la colonne de chaque élément de Dalle.
3. La mesure spatiale de chaque Dalle.
4. Une URL d'image pour chaque Dalle.

```
1  <TileGrid columns="4" rows="4">
2      <CenterContext SRS="WGS-84">
3       <CenterPoint>
4        <pos>41.002 -72.000896</pos>
5       </CenterPoint>
6       <Radius unit="KM">1.0</Radius>
7      </CenterContext>
8       <Tile col="0" row="0">
9        <Map>
10        <Content height="300" format="GIF"
11 width="300">
12 <URL>http://ws.deCarta.com:8080/openls/image/TILE?
13 LLMIN=41.00361987041037,-72.00447335745504
14 &LLMAX=41.00469978401728,-72.00304241447303
15 &WIDTH=300&HEIGHT=300&FORMAT=GIF&
16 CLIENTNAME=someclient&SESSIONID=999&
17 N=0&E=0</URL>
18       </Content>
19        <BBoxContext>
20           <pos>41.00361987041037 -
21 72.00447335745504</pos>
22           <pos>41.00469978401728 -
23 72.00304241447303</pos>
24        </BBoxContext>
25       </ns1:Map>
      </ns1:Tile>
```

Listing. 2. A TileGrid element taken from a PortrayMapResponse. Only the first Tile in the TileGrid is shown.

Le listing ci-dessus montre un élément de Dalle simple à l'intérieur du TileGrid a retourné dans la PortrayMapResponse. L'URL pour chaque image de Dalle n'est pas une URL statique, mais plutôt un jeu de paramètres qui permettent à la Dalle d'être rendus "just in time". Le client peut ignorer tous les paramètres en question dans l'URL, à part les deux derniers paramètres. Ces paramètres sont mis en évidence dans le listing ci-dessus et sont décrits comme les paramètres de mouvement vers l'Est et vers le Nord. Ces deux paramètres sont essentiels à l'architecture de déplacement dans la carte.

2.1.2.4 Déplacement dans la carte

L'architecture de JavaScript pour le déplacement des cartes utilise des éléments HTML « DIV », la TileGrid est disposé dans une DIV, contenu dans un autre élément DIV ; nous les appellerons respectivement DIV intérieure (Inner DIV) et DIV extérieure (Outer DIV). Le déplacement se fait lors de la détection des événements JavaScript 'onmousedown' et 'onmousemove' en alternance avec les attributs 'top' et 'left' des DIV de la feuille de style CSS.

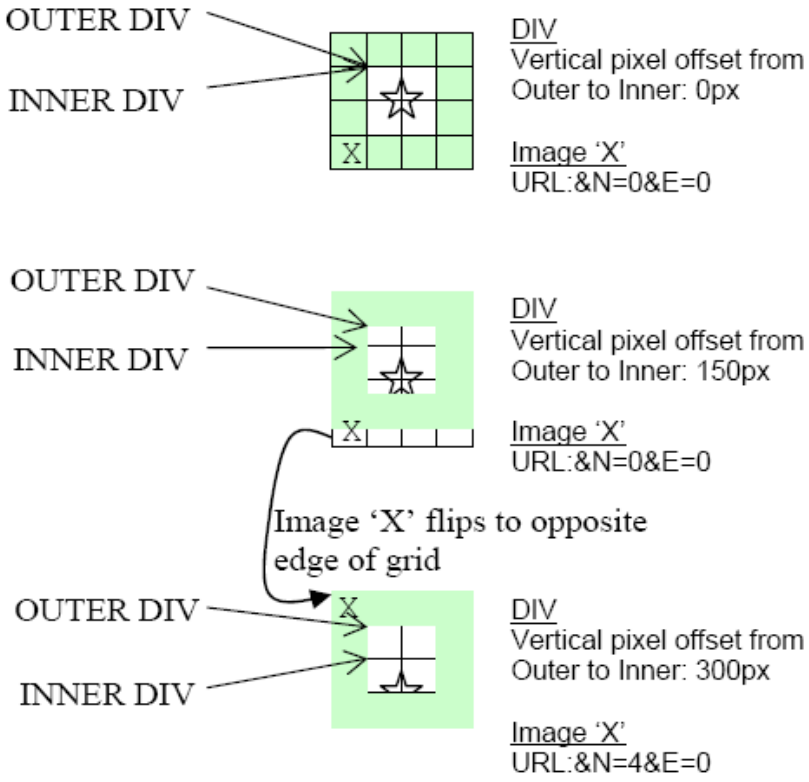


Fig. 3 – Sequence of user dragging map vertically downwards

La figure ci-dessus montre une séquence, dans laquelle l'utilisateur effectue un déplacement en bas de page dans la carte. L'étoile représente le centre de l'adresse (CenterAddress) de la grille de dalles (TileGrid). La DIV intérieure encadre la DIV extérieure, pour créer une zone visible centrée aux 4 dalles intérieures.

L'attribut 'overflow' de la feuille de style est mis à 'hidden' pour la DIV extérieure, donc elle est cachée. Il faut bien noter que cette dernière a un index supérieur à celui de la DIV intérieure.

Dans la première capture, la carte n'est pas déplacée, donc la DIV intérieure a un offset vertical égal à zéro par rapport à sa DIV extérieure.

L'image mentionnée par X et qui a 300x300 pixels comme taille, a une URL avec des paramètres Nord et Sud égales à zéro. ' X' à 600 pixels comme offset initial vertical avec la DIV.

La deuxième capture montre que l'utilisateur a déplacé la carte de 150 pixels vers le bas. Les offsets des images avec la DIV intérieure n'ont pas été changés, juste la DIV intérieure a été déplacée relativement avec la DIV extérieure.

Dans la troisième capture, la DIV intérieure a été déplacé de 300 pixels vers le bas, à ce moment là, l'image 'X' est déplacée du bas de la grille vers le haut, en obtenant un nouvel offset

égale à -300 relative a la DIV intérieure, de la même façon, les URL des images obtiennent des valeurs de Nord égales à 4, cette valeur de 4 représente la hauteur de la grille(4 dalles) .

2.1.2.5 Zoom

Le client ne doit pas calculer la lat/lng du déplacement de la carte avant le zoom. La requête de TileGrid "PortrayMapRequest" permet au client de spécifier le nombre de dalle avec lequel la carte a été déplacée, dans n'importe quelle direction. Cela est accompli par l'inclusion d'éléments Pan dans la requête TileGrid "PortrayMapRequest".

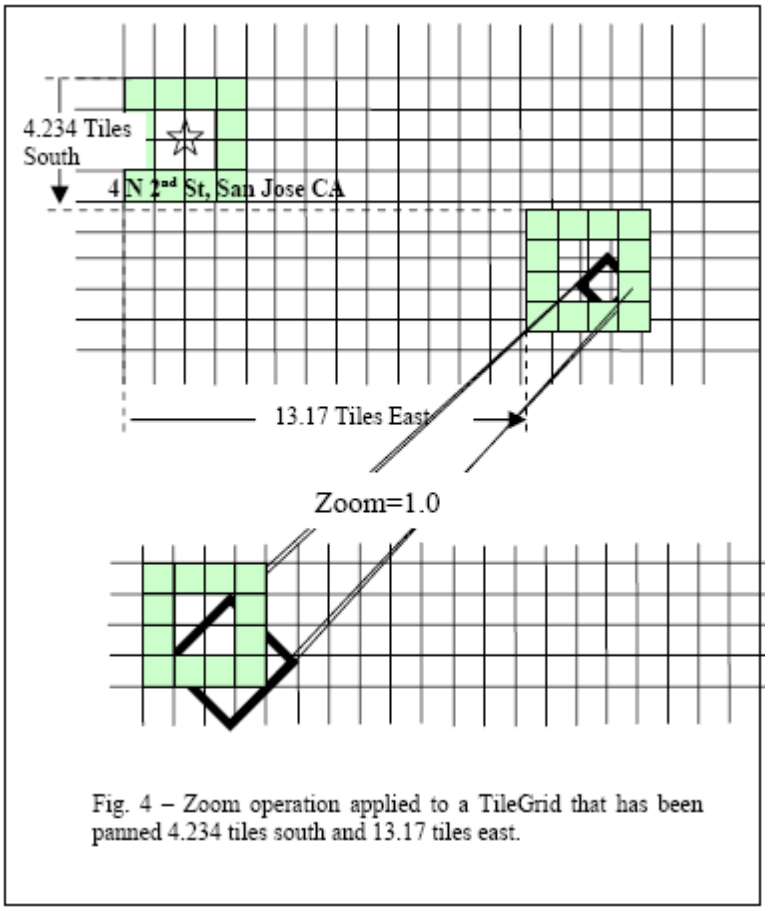


Fig. 4 – Zoom operation applied to a TileGrid that has been panned 4.234 tiles south and 13.17 tiles east.

La figure ci-dessus montre un TileGrid que l'utilisateur a déplacé de 4.234 dalles vers le Sud et 13.17 dalles vers l'ouest. Une opération de zoom est alors appliquée au TileGrid, aboutissant à un nouveau TileGrid retourné par le serveur.

```
1   <PortrayMapRequest>
2     <Output width="300" height="300" zoom="1.0">
3       <CenterContext SRS="WGS-84">
4         <CenterPoint>
5           <pos>41.002 -72.000896</pos>
6         </CenterPoint>
7         <Radius unit="KM">1.0</Radius>
8       </CenterContext>
9       <TileGrid rows="4" columns="4">
10        <Pan direction="E" numTiles="13.17"/>
11        <Pan direction="S" numTiles="4.234"/>
12      </TileGrid>
13    </Output>
14  </PortrayMapRequest>
```

Listing. 3. A PortrayMapRequest combining a zoom on top of a panned TileGrid

Le listing 3 ci-dessus montre une PortrayMapRequest qui contient une requête de zoom de facteur de 1.0. La Ligne 2 contient l'attribut zoom. La requête inclut des informations sur comment la TileGrid courante a été déplacé. Les lignes 10 et 11 montrent les éléments Pan discrets qui disent au serveur de combien d'utilisateur a déplacé la carte. Le serveur a besoin de ces informations parce qu'il doit appliquer le facteur de zoom à l'emplacement final ou l'utilisateur a déplacé la carte.

Notez que tandis que le client pourrait avoir fourni le CenterAdress originale, dans ce cas le client a utilisé le CenterContext fourni dans la réponse montrée dans le listing 2.

C'est montré dans les lignes de 3 à 8 dans le listing 3. En utilisant la lat/lng du CenterContext fourni par le serveur avec la réponse du TileGrid initiale, geocoding est seulement exécuté sur le positionnement initial du TileGrid, plutôt que sur chaque opération de zoom suivante.

2.1.2.6 Itinéraires et Overlays

Les DDS Web Services supportent le rendu des Overlays avec la balise TileGrid.
Les Overlays peuvent être des itinéraires géométriques, POIs, texte, etc.
Un avantage du rendu des dalles dynamique, par opposition aux dalles pré-rendues, consiste en ce que tous les itinéraires et d'autres overlays font partie de l'image de la dalle. Dans les systèmes qui utilisent les dalles pré-rendues, des itinéraires doivent être rendues comme un PNG avec un fond transparent et mis sur des dalles dans le client JavaScript, augmentant la complexité du système.

Les overlays sont spécifiés comme une partie de la requête initiale du TileGrid et sont associéent aux dalles via la sessionID. Si on ne fournit pas de sessionID dans la requête, le serveur rendra une unique sessionID. L'URL pour des dalles individuelles inclura cet sessionID. Si une application fournit un sessionID, l'application devra choisir une seesionID unique pour chaque utilisateur afin de ne pas "se mélanger à" des itinéraires et d'autre overlays sur la carte d'un utilisateur différent.

```
1   <PortrayMapRequest fitOverlays="true">
2     <Output height="400" width="400">
3       <CenterAddress>
4         <Radius unit="KM">5</Radius>
5         <Address countryCode="US">
6           <freeFormAddress>300 Second Ave,
7   11111</freeFormAddress>
8         </Address>
9       </CenterAddress>
10      <TileGrid rows="3" columns="3"/>
11    </Output>
12    <Overlay>
13      <RouteGeometry>
14       <LineString ">
15        <pos>41.0039 -72.003</pos>
16        <pos>41.003 -72.003</pos>
17        <pos>41.003 -72.003</pos>
18        <pos>41.002 -72.003</pos>
19        <pos>41.002 -72.003</pos>
20        <pos>41.001 -72.003</pos>
21        <pos>41.001 -72.002</pos>
22        <pos>41.001 -72.002</pos>
23        <pos>41.001 -72.001</pos>
24        <pos>41.001 -72.001</pos>
25        <pos>41.001 -72.00011</pos>
26       </LineString>
27      </RouteGeometry>
28    </Overlay>
29  </PortrayMapRequest>
```

Listing. 4. A request for a TileGrid with a route drawn on the tiles.

Le listing 4 montre e PortrayMapRequest qui inclut un TileGrid et RouteGeometry pour recouvrir une dalle. La ligne 1 montre l'attribut de FitOverlays. Cet attribut fera l'extension spatiale du réseau de dalles pour "s'étirer" pour couvrir tous les overlays. Les lignes 13-27 montre la géométrie d'itinéraire. Dans une application AJAX typique, une DeterminRouteRequest initiale serait utilisée pour obtenir l'itinéraire géométrique et la PortrayMapRequest suivant serait faite pour recouvrir la géométrie, comme indiqué dans la figure 4. Les opérations de zoom doivent aussi inclure la géométrie d'itinéraire.

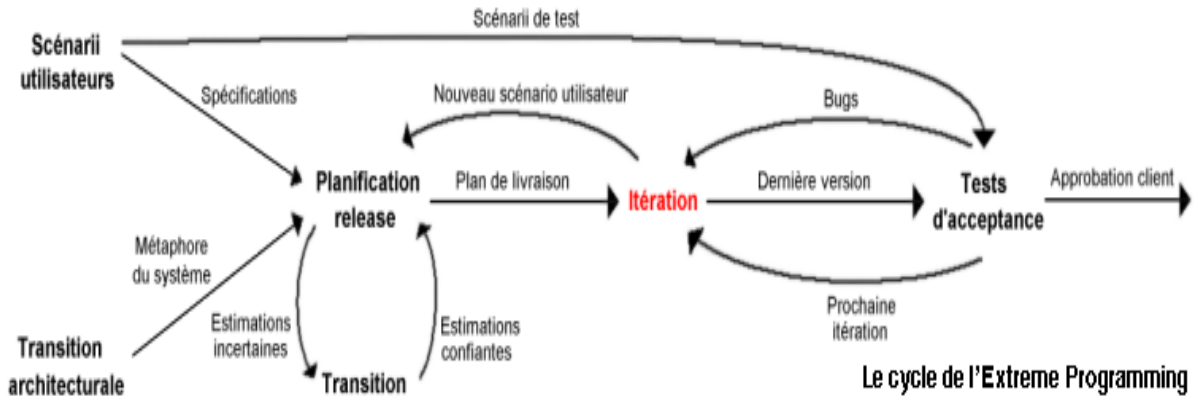
3.CAHIER DES CHARGES

3.1 Présentation de la méthode de travail

Sur la demande de notre encadrant, la méthode de l'extrême programming sera adoptée comme processus de développement tout au long du projet.

Ce qui présuppose un partage des tâches par binôme. Le CLIENT dans ce processus sera notre encadrant. C'est le CLIENT qui va piloter le projet, voir ce qui est nécessaire et ce qui ne l'est pas. *(Changement d'objectifs, rajout ou élimination de certaines fonctionnalités, etc.)*

Voici les principaux éléments du fonctionnement de notre projet :



- **Cycles itératifs pilotés par le CLIENT** : Le projet progresse au rythme d'itérations très courtes (deux semaines environ), dont le contenu fonctionnel est déterminé par le client (l'encadrant).

- **Travail d'équipe auto-organisé** : L'ÉQUIPE travaille en étroite collaboration. Nous organisons nous-mêmes notre travail sous la direction de l'encadrant. L'ÉQUIPE intervient sur l'ensemble du code, travaille systématiquement en binômes et synchronise son développements tout le temps.

- **Programmation pilotée par les tests** : Nous écrivons des tests automatiques pour chaque portion de code conçu et nous nous appuyons sur ces tests pour affiner et améliorer sans cesse la conception de l'application sans craindre de régression.

Remarque : De part la description de notre méthode de travail, la répartition des tâches entre les différents membres du groupe n'est pas définitive.

3.2 Communication

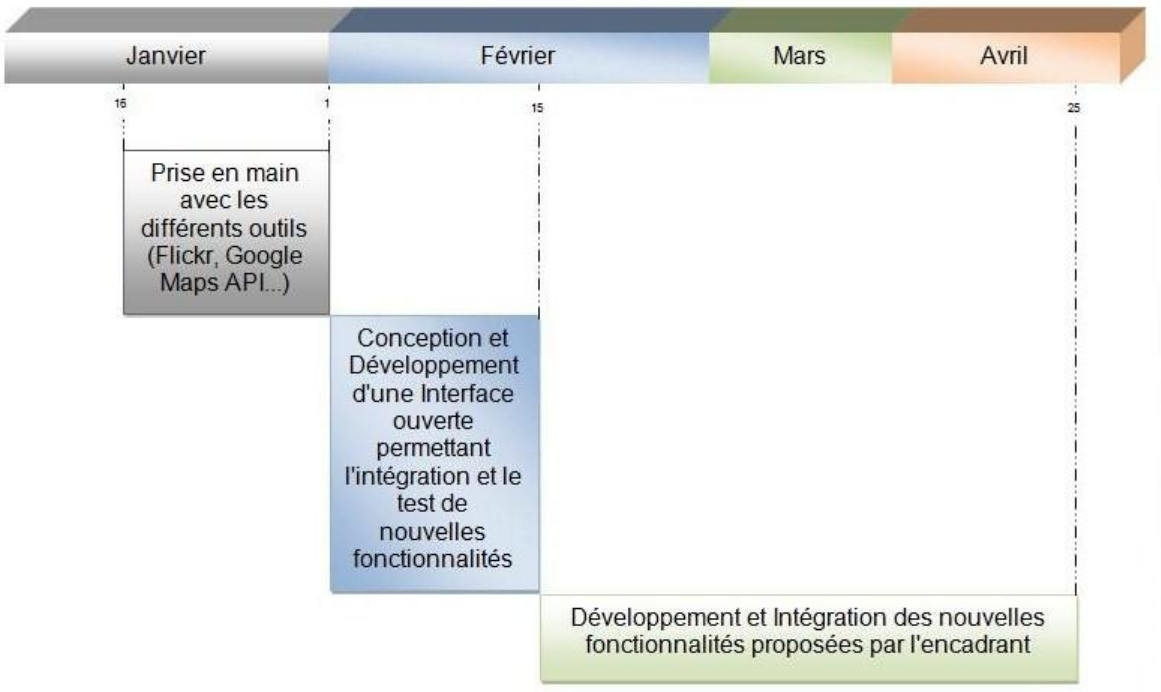
Nous prévoyons de rencontrer notre encadrant, Mr Delort, toutes les semaines. À chaque rendez-vous, nous lui présenterons un rapport comportant le travail effectué et une démonstration. Au cours de ces réunions, notre encadrant nous apportera des informations complémentaires pour la continuité du projet.

Nous utiliserons aussi les outils suivants pour la communication quotidienne :
Subversion (partage des sources avec GoogleCode, membres du groupe)
Skype (conférence, membres du groupe)
Liste de diffusion (avec GoogleGroups, membres du groupe et encadrant)

Des rendez-vous seront organisés entre les membres du groupe tous les vendredis matin pour faire part des difficultés éventuelles.

3.3 Présentation du planning

3.3.1 Diagramme de Gantt



4.STRUCTURE DU SITE

4.1 Hiérarchie du Géo-Service

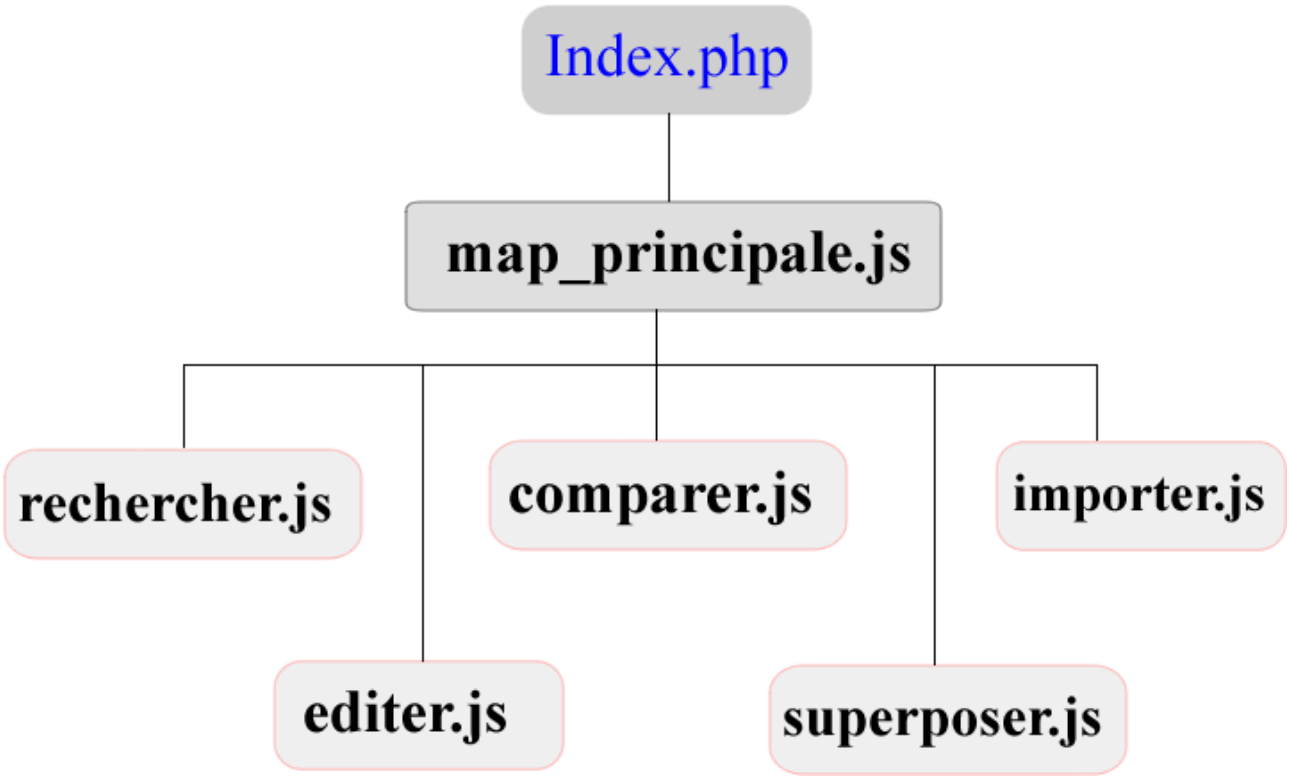


Figure 4.1 : Hiérarchie du Géo-Service

4.2 Structure du Géo-Service

La structure du géo-service a été réalisée pour respecter les contraintes du cahier des charges sur la visualisation des résultats des cartes géographiques Google Maps.

La plupart des Géo-Services proposent de manipuler une carte à la fois selon le service traité.
La structure que nous avons développée lors de notre mashup de l'API Google Maps nous permet facilement de manipuler plusieurs cartes et d'intégrer d'autre service web selon leur catégorie.

Nous identifions 3 types de catégories pour les services web:

- catégorie de recherche (ex: flickr...)
- catégorie d'édition
- catégorie de comparaison

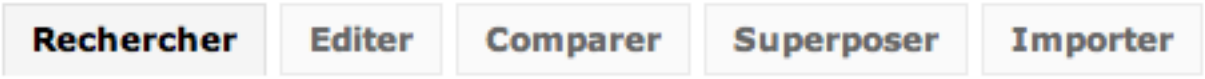


Figure 4.2 : Barre de navigation

Pour organiser la répartition des fonctions à développer entre les membres du groupe, il a été décidé que nous devions avoir un fichier par fonctionnalité.

- map_principale.js
  Comporte les fonctions communes aux différentes fonctionnalités.
- map_rechercher.js
  Comporte les fonctions et services web en rapport à la recherche de résultat.
- map_editer.js
  Comporte les fonctions et services web nécessaire pour l'édition des cartes.
- map_comparer.js
  Comporte les fonctions et services web nécessaire pour comparer des cartes.
- map_superposer.js
  Comporte les fonctions et services web nécessaire pour superposer des cartes.
- map_importer.js
  Comporte les fonctions qui intègrent à notre projet des fichiers KML.

4.3 Structure de donnée

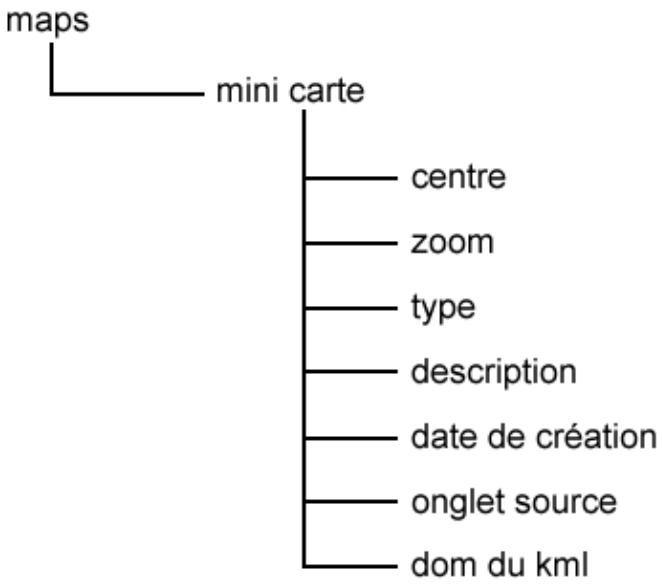


Figure 4.3 : Structure de donnée de la liste des cartes

Nous avons développé une structure de donnée commune aux différentes fonctionnalités qui récupère les informations des différentes cartes. Cette structure compose en grande partie la liste des cartes.
Pour simplifier la manipulation des éléments relatifs aux cartes, nous avons utilisé une structure d'arbre DOM semblable à la DTD d'un fichier KML.

5.FONCTIONNALITES

5.1 Cas d'utilisations

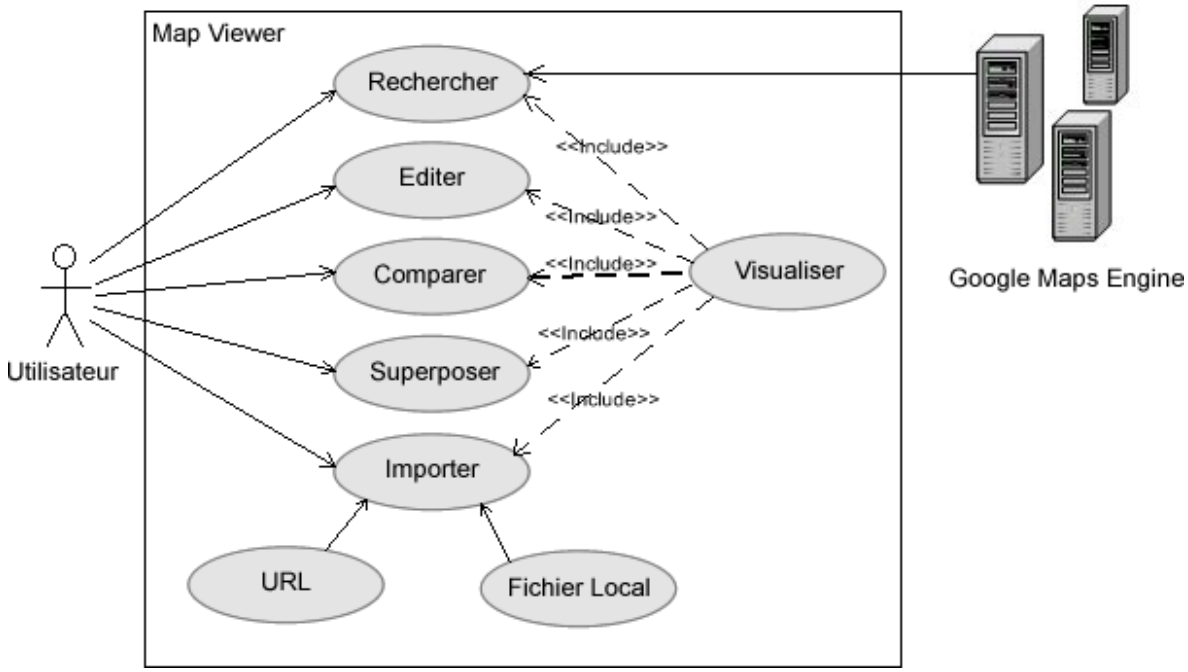


Figure 5.1 : Cas d'utilisation du projet Maps Viewer

5.2 Présentation des différentes fonctionnalités

5.2.1 Fonctionnalités communes

5.2.1.1 La liste des cartes

La liste des cartes est la fonctionnalité qui permet de stoker instantanément les différentes cartes affichées dans les différents onglets, pour ensuite, les réafficher au besoin de l'utilisateur.

Structure

La liste des cartes est une DIV qui se situe à gauche des onglets, et qui contient des mini-cartes, ces mini-cartes, sont des DIV créés dynamiquement à chaque fois qu'une carte est ajoutée à la liste des cartes; chaque DIV créé et représenté par un nom qui est le nom de la carte, et une image qui est la capture de la carte au moment de l'ajout.

Figure 5.1: La liste des cartes

A chaque DIV, est associée une structure qui contient toutes les informations nécessaires sur la carte, et qui permettent de la réafficher plus tard, cette structure est la même pour tous les div, et elle contient :

- ***Centre de la carte :*** la latitude est la longitude de la carte au moment de l'ajout.
- ***Niveau de zoom :*** le niveau de zoom au moment de l'ajout.
- ***Type de la carte :*** plan, satellite ou mixte.
- ***Description :*** description sur la carte saisie par l'utilisateur.
- ***Date de création :*** la date au moment de l'ajout.
- ***Onglet source :*** l'onglet où la carte a été ajoutée.
- ***Dom :*** structure spécifique aux cartes (format KML) :
    - ***Nom de la carte :*** nom saisi par l'utilisateur, ou le nom du fichier KML.
    - ***Marqueurs :*** tous les marqueurs la carte.
    - ***Polylines :*** toutes les lignes et les multi lignes de la carte.
    - ***Polygones :*** tous les Polygones la carte.

Fonctionnement :

Le fonctionnement de la liste des cartes est le suivant :

- **Au moment de l'ajout :**

Lors de l'ajout d'une carte à la liste des cartes, une boite de dialogue s'affiche, comme le montre la figure suivante.

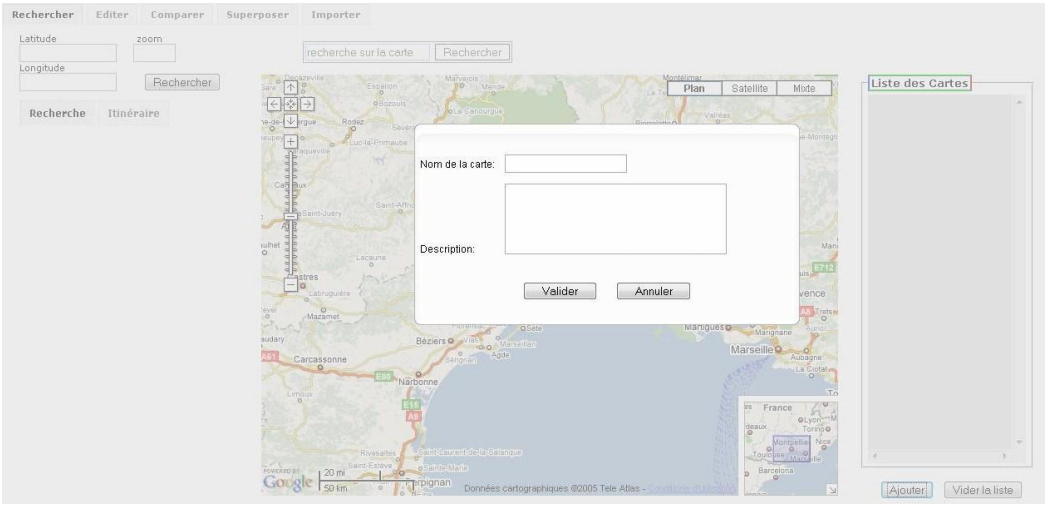


Figure : Ajout à la liste des carte

L'utilisateur doit saisir le nom de la carte et la description, une fois qu'il a fini :

- Un nouveau div sera crée :
  var nouveauDiv = document.createElement("div");
- La carte statique sera créée :
  var image = document.createElement("img") ;
  *image.src = lien vers l'image. (*Utilisation de Google Static Maps API)
- Associer le nom entré par l'utilisateur à la carte s'il s'agit d'une nouvelle carte, ou le chercher depuis le fichier KML s'il s'agit d'une carte importée :

```
if(vueCourante !="mapImporter"){
        imageText = nomCarte;
    }else{
        imageText = dom.documentElement.getElementsByTagName("name")[0].textContent;
    }
```

- Stoker le rang du div dans un attribut dans le div lui-même:
  nouveauDiv.setAttribute("indice",compteur);
  ce rang est l'indice de la structure du div dans le tableau des div, ce rang est indispensable pour gérer l'événement associé au div.
- Ajouter le nom et l'image de la carte au div créé, ensuite, ajouter le div créé au div de la liste des cartes :
  nouveauDiv.appendChild(document.createTextNode(imageText));
  nouveauDiv.appendChild(image);

document.getElementById(divListeCartes).appendChild(nouveauDiv);

- Sauvegarder les informations sur la carte dans un tableau de tructure:

var Map = new Object();
Map.Center = mapCourante[vueCourante].getCenter();
Map.Zoom = mapCourante[vueCourante].getZoom();
Map.Type = mapCourante[vueCourante].getCurrentMapType();
Map.OngletSource = vueCourante;
Map.DateCreation = CreationDate;
Map.description = descriptionCarte;

dom.documentElement.getElementsByTagName("name")[0].innerHTML = nomCarte ;
Map.Dom = dom;
maps[compteur] = Map;

Google Static Maps API

Sachant que la carte n'est pas une image statique, mais un élément graphique interactif et déplaçable, alors, pour travailler un peu l'IHM de notre application, on a utilisé l'API Google Static Maps pour avoir des images statiques des cartes.

L'API Google Static Maps retourne une image (GIF, PNG ou JPEG) en réponse à une requête HTTP via une URL. Cette URL est composée de plusieurs paramètres. Certains paramètres sont obligatoires, alors qu'il y a d'autres qui sont optionnels, tous les paramètres sont séparés par &.

L'URL doit avoir la forme suivante :

*http://maps.google.*com/staticmap?paramètres

La liste de paramètres:

- Centre :
- Zoom :
- Taille :
- Type :
- Marqueurs :
- Clé :

Exemple d'une URL :

http://maps.google.com/staticmap?center=40.714728,73.998672&zoom=12&size=400x400&maptype=mobile&markers=40.702147,-74.015794&key=MAPS_API_KEY

Au moment du click sur une mini-carte :

L'événement est recupéré par la ligne :
nouveauDiv.onclick=function (event) {}

Récupérer le rang de la mini-carte dans la liste des cartes :

var i = this.getAttribute('indice');

Après le click, le but est d'afficher la carte correspondante, mais selon l'onglet où le click est fait, l'exécution se diffère. En général, à chaque click, on passe à une fonction, le div où il faut afficher la carte, le centre de la carte, le niveau de zoom, le type et le dom qui contient les marqueurs, lignes et polygones :

loadmap(DivMapCourante,maps[i].Center,maps[i].Zoom,maps[i].Type,maps[i].Dom);

et la fonction *loadmap(div,centre,zoom,type,dom) aura la structure suivante*

```
function loadmap(div,centre,zoom,type,dom) {
    if (GBrowserIsCompatible()) {
        var carte = new GMap2(document.getElementById(div));
        carte.setCenter(centre,zoom);
        carte.setMapType(type);
         afficheDOM(dom,carte);
        }
    }
```

La fonction afficheDOM() affiche les marqueurs, lignes et polygones, cette fonction sera détaillée dans ce qui suit.

Au moment du survol sur la mini-carte avec la souris *:*

Lors du survol avec la souris, on affiche des informations sur la carte, par exemple, la description, la date, l'onglet source…etc.

L'événement est récupéré par :

nouveauDiv.onmouseover= function(event) {}

Même chose pour le click, on récupère le rang de la mini-carte dans la liste des cartes pour qu'on puisse extraire les informations depuis le tableau des cartes:

*var j = this.*getAttribute*('indice');*

L'affichage des informations se fait en appelant une fonction qui affiche un div qui va contenir les informations, ce dernier est affiché aux mêmes coordonnées de la souris, juste un petit décalage, pour la visibilité.

Synchronisation de la liste des cartes avec les onglets :

Pour que la liste des cartes soit utilisable par tous les onglets, il faut un mécanisme de synchronisation et d'adaptation de tel sorte que la modification de la liste dans un onglet, soit appliquée dans les autres onglets, même chose pour l'ajout ou la suppression, et bien évidement le réaffichage des cartes, ce mécanisme est basé sur plusieurs éléments :

- *tabDiv = new Array ("listeRechercher","listeEditer","listeComparer","listeSuperpose") :* Tableau qui contient les id des div des listes des cartes pour chaque onglet.

- *DivMapCourante :* variable qui va contenir l'id du div où la carte est affichée,

- vueCourante : variable qui contient la vue courante, à chaque fois qu'on bascule d'un onglet à un autre, elle va prendre comme valeur, le nom de l'onglet en premier plan.
  Par exemple, si on bascule vers l'onglet superposer, la ligne suivante va être exécutée:
  vueCourante = "mapSuperpose"

- *var mapCourante = new Object () ;* objet qui va contenir l'objet map c.-à-d. le résultat de la ligne : map = new GMap2(document.getElementById("div_map_rechercher"));
  En réalité en va l'utiliser comme un tableau des objets map, par exemple, la ligne suivante *: mapCourante["mapRecherche"],* contient l'objet map de l'onglet rechercher, et cet objet est initialisé dans la fonction qui affiche la carte par cette ligne :
  mapCourante["mapRecherche"] = map;

Au moment de l'ajout, et puisque la liste des cartes se figure dans les onglets : rechercher, éditer, comparer et superposer. L'ajout se fait en boucle et dans les quatre onglets comme suit :

```
function AjouterCarte(Dom){
        for(i=0; i<4;i++){
                mis_à_jours_Listes(tabDiv[i],Dom);
                }
                compteur++;
}
```

- La fonction *mis_à_jours_Listes(tabDiv[i],Dom*) fait exactement l'ajout déjà expliqué en haut.
- tabDiv[i] : contient les id des div de la liste des cartes pour chaque onglet.
- Dom: la structure de données qui contient les marqueurs, lignes et polygones, elle sera détaillée dans ce qui suit.

Effacement de la liste :

L'effacement de la liste se fait lors du click sur le bouton Vider_La_Liste, une fenêtre de dialogue s'ouvre pour demander une confirmation.

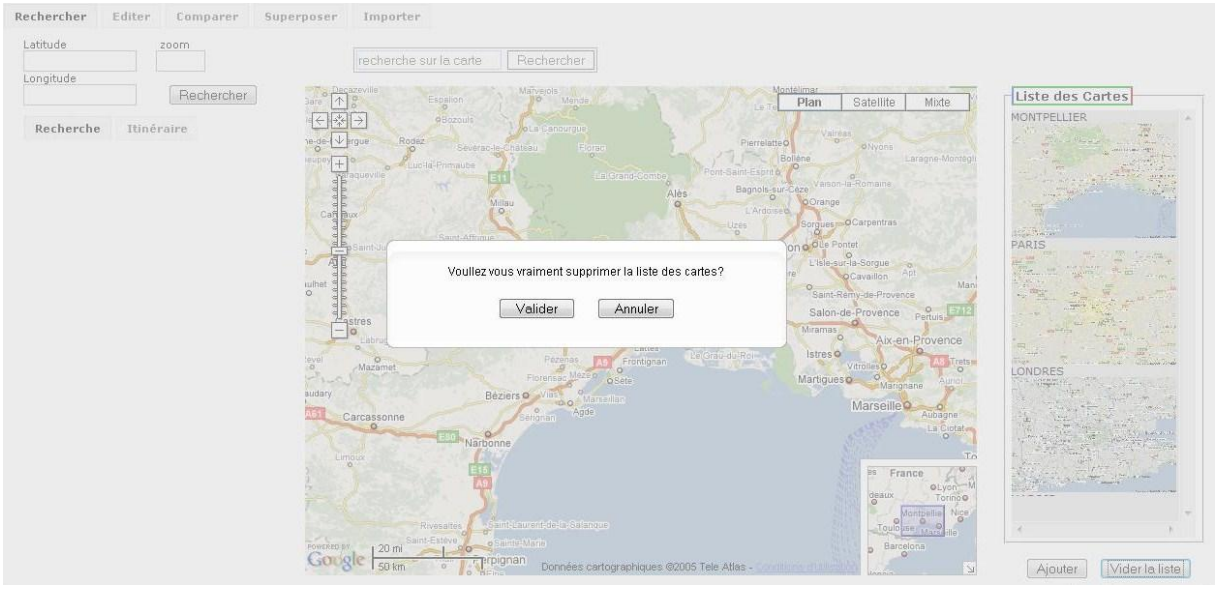


Figure : Effacement de la liste des cartes

Après la confirmation, la liste sera vidée avec cette ligne :

document.getElementById(tabDiv[i]).innerHTML="";

Ensuite, il faut remettre le compteur à zéro.

Effacement d'une mini-carte :

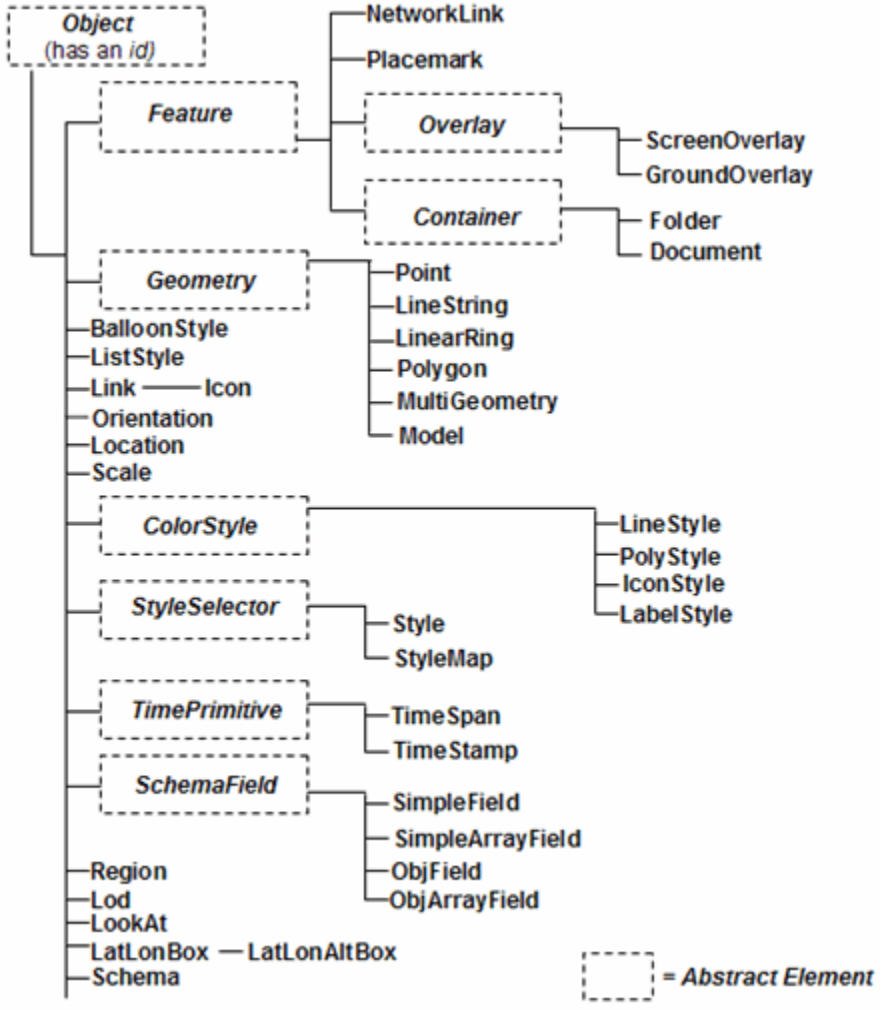
Lors d'un click droit sur une mini-carte, une fenêtre s'ouvre, l'utilisateur confirme la suppression, et la mini-carte sera effacée :

document.getElementById("tabDiv[i]").removeChild(this);

5.2.1.2 KML

Les KML sont des fichiers XML qui respectent une certaine DTD représentant les éléments pouvant être contenue sur une carte Google Maps ou Google Earth.

L'arborescence des éléments KML est illustrée ci-dessous. Sur ce schéma, sur chacune des branches de l'arborescence, les éléments se trouvant à droite sont des extensions des éléments se trouvant à leur gauche. Par exemple, Placemark (repère) est une extension de Feature (fonctionnalité). Un repère contient tous les éléments appartenant à Feature et y ajoute certains éléments spécifiques à Placemark.



Remarque: L'éditeur de carte de Google Maps ne propose plus d'exporter les cartes que vous éditez. Pour pouvoir exporter vos cartes, vous devez récupérer l'URL de la carte à partir du lien "Obtenir l'URL e cette page" et ajouter la commande "&output=kml"

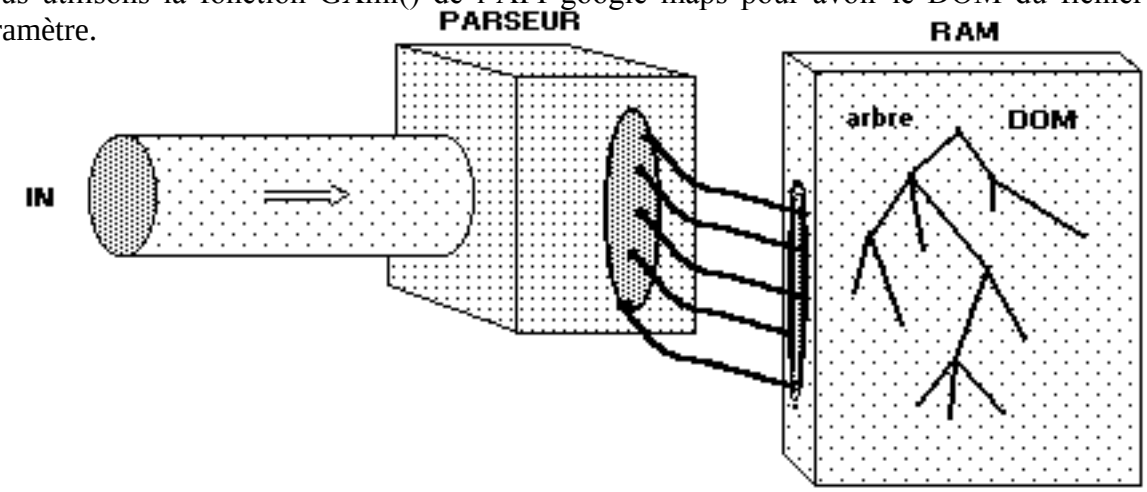
exemple:
http://maps.google.fr/maps?q=montpellier&ie=UTF8&oe=utf-8&client=firefox-a&z=12&iwloc=addr&output=kml

5.2.1.3 KML Virtuel

l'API de google maps ne nous propose pas de fonction pour récupérer les éléments contenu dans les cartes.

La solution que nous avons adopté, a été de charger le KML en mémoire du navigateur.
De ce fait, nous travaillons sur 2 plans. Nous ajoutons les éléments à la carte de google maps et nous mettons à jour en même temps le KML virtuel qui représente le DOM du fichier KML chargé.

Nous utilisons la fonction GXml() de l'API google maps pour avoir le DOM du fichier mis en paramètre.



5.2.1.4 AfficherDOM()

La fonction afficherDOM() tout comme la fonction GGeoXML() de l'API de Google Maps, affiche tout les éléments de l'objet contenu en paramètre. À la seul différence que GGeoXML() prends en paramètre l'url d'un fichier et que afficherDOM() un arbre DOM respectant une partie de la grammaire du KML présenté par Google Maps.

Fonctionnement

La fonction afficherDOM() est constitué de bloc conditionnels qui ajoute à l'Overlays de la carte courante l'élément correspondant aux différentes balises.

Les éléments géographiques (marqueurs, polygones, polylines) contenue dans une carte Google Maps sont constitués:

- nom
- descriptif au format HTML
- coordonnées (latitude, longitude)
- style (apparence visuel)

Utilisation

Cette fonction est utilisée à chaque fois que nous demandons l'affichage d'une carte contenu dans la liste des cartes.

5.3 Rechercher:

L'onglet rechercher est le point d'entrée de notre application, des que la page se charge, l'onglet rechercher apparaît, la figure ci-dessous montre son l'interface

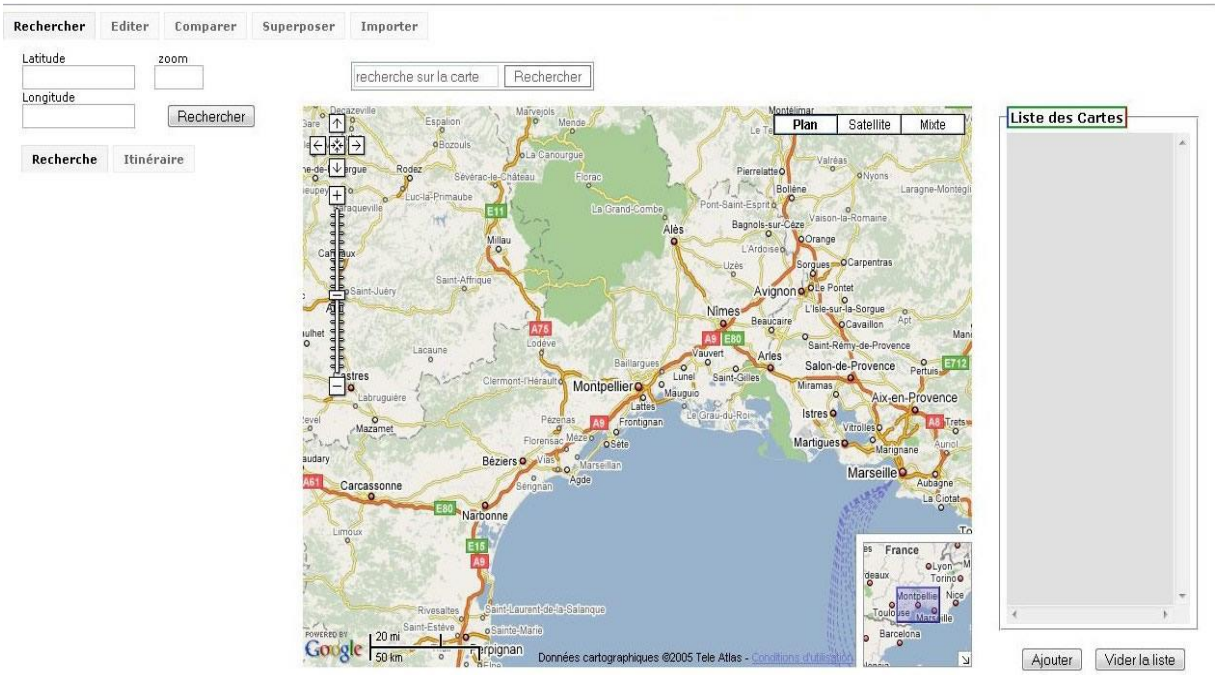


Figure : Onglet Rechercher

L'onglet rechercher à comme principales rôles, de faire des recherches, et donner des itinéraires.

1.1. La Recherche :

Pour la recherche, on a utilisé l'API GoogleMap pour afficher les résultats de recherche dans un div externe du div de la carte, et aussi la barre de recherche ou ce que Google appel « Control » est à l'extérieur de la carte,

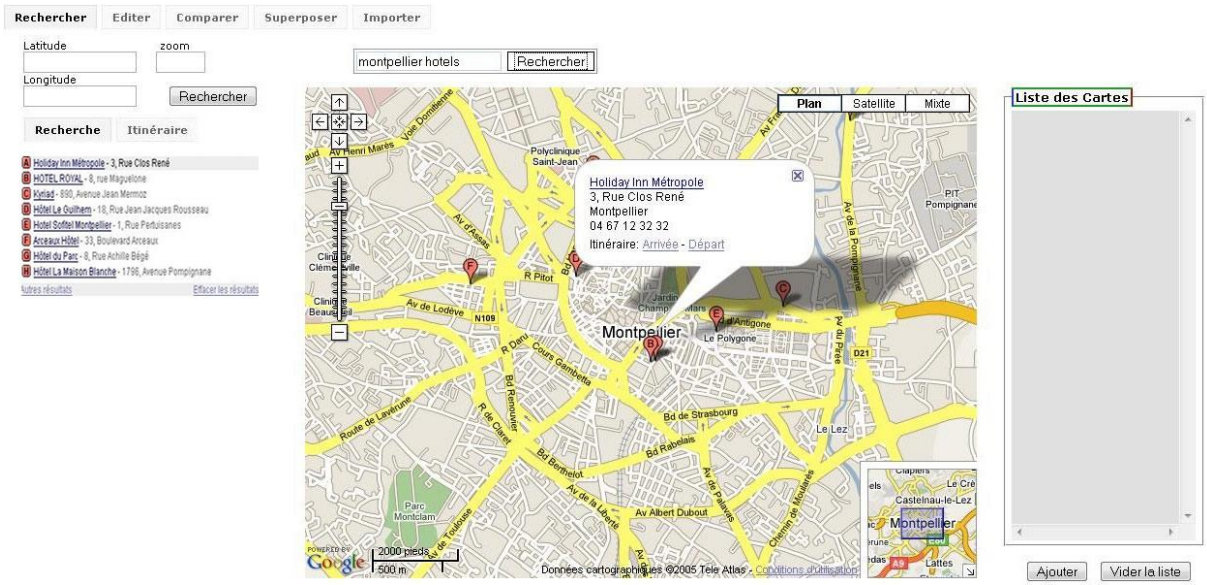


Figure : Résultats de recherche

Le listing suivant montre le code.

```
map.addControl(new google.maps.LocalSearch(
      {
        resultList : document.getElementById("results")
       }),
              map.addControl(new google.maps.LocalSearch(options));
                 );
```

1.2. Le calcul d'itinéraire:

Le calcul d'itinéraire se fait on utilisant l'objet -Gdirections- qui est initialisé avec l'objet map, et le div ou on va afficher le résultat de la requête passée à GDirections.

Le résultat sera un tracé d'itinéraire entre l'adresse de départ et l'adresse d'arrivée. Et un guide dans un div. la figure ci-dessous Montre le résultat.

```
gdir = new GDirections(map, document.getElementById("DIVRésulats"));
GEvent.addListener(gdir, "load", onGDirectionsLoad);
GEvent.addListener(gdir, "error", handleErrors);
```

Il faut ajouter des écouteurs sur la carte, et les associé à l'objet GDirections, pour pouvoir naviguer sur la carte et voir en détails l'itinéraire.

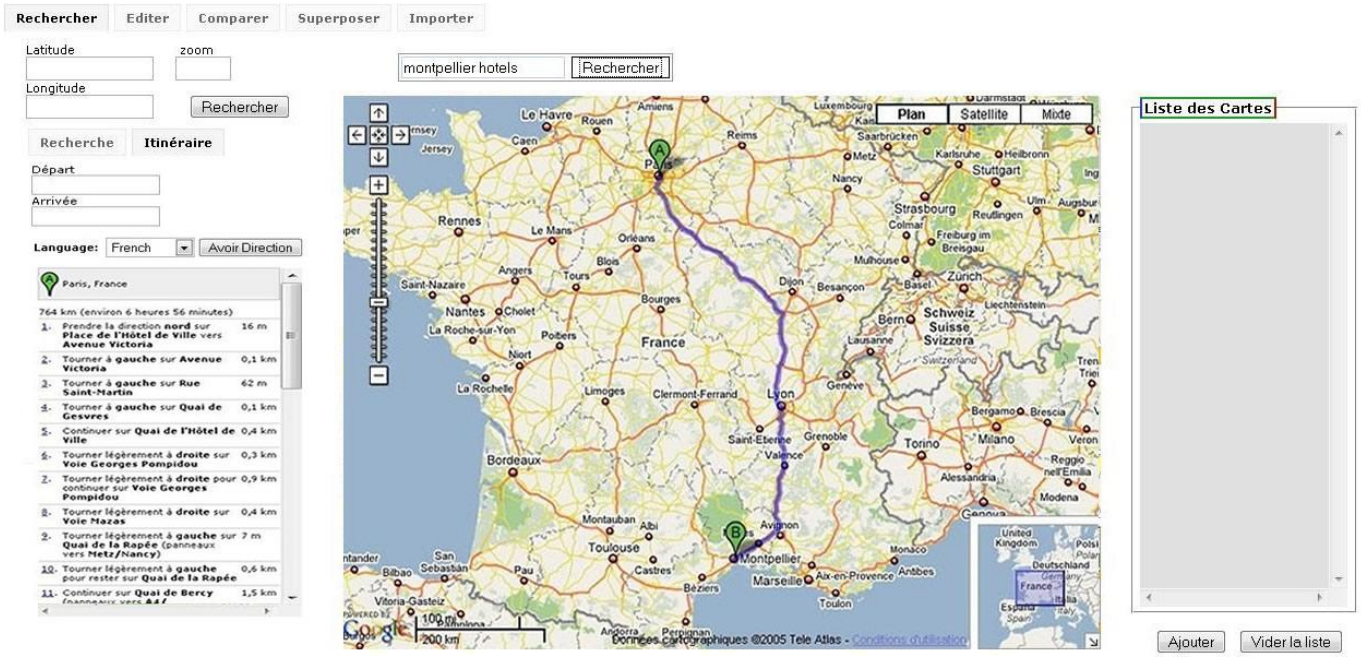


Figure: Calcule d'itinéraire.

Des que l'utilisateur à choisi l'adresse de départ et d'arrivée, et valider, la fonction setDirections() sera appelée, le Listing suivant montre le code :

```
function setDirections(fromAddress, toAddress, locale) {
gdir.load("from: " + fromAddress + " to: " + toAddress,
{ "locale": locale });
}
```

Les deux premiers paramètres sont l'adresse de départ et d'arrivée, et le dernier c'est la langue choisie par l'utilisateur.

5.3.1 Editer:

L'onglet éditer représente un axe important dans le site, car il interagit activement avec les onglets : rechercher, superposer et importer.

5.3.1.1 Présentation de l'interface:

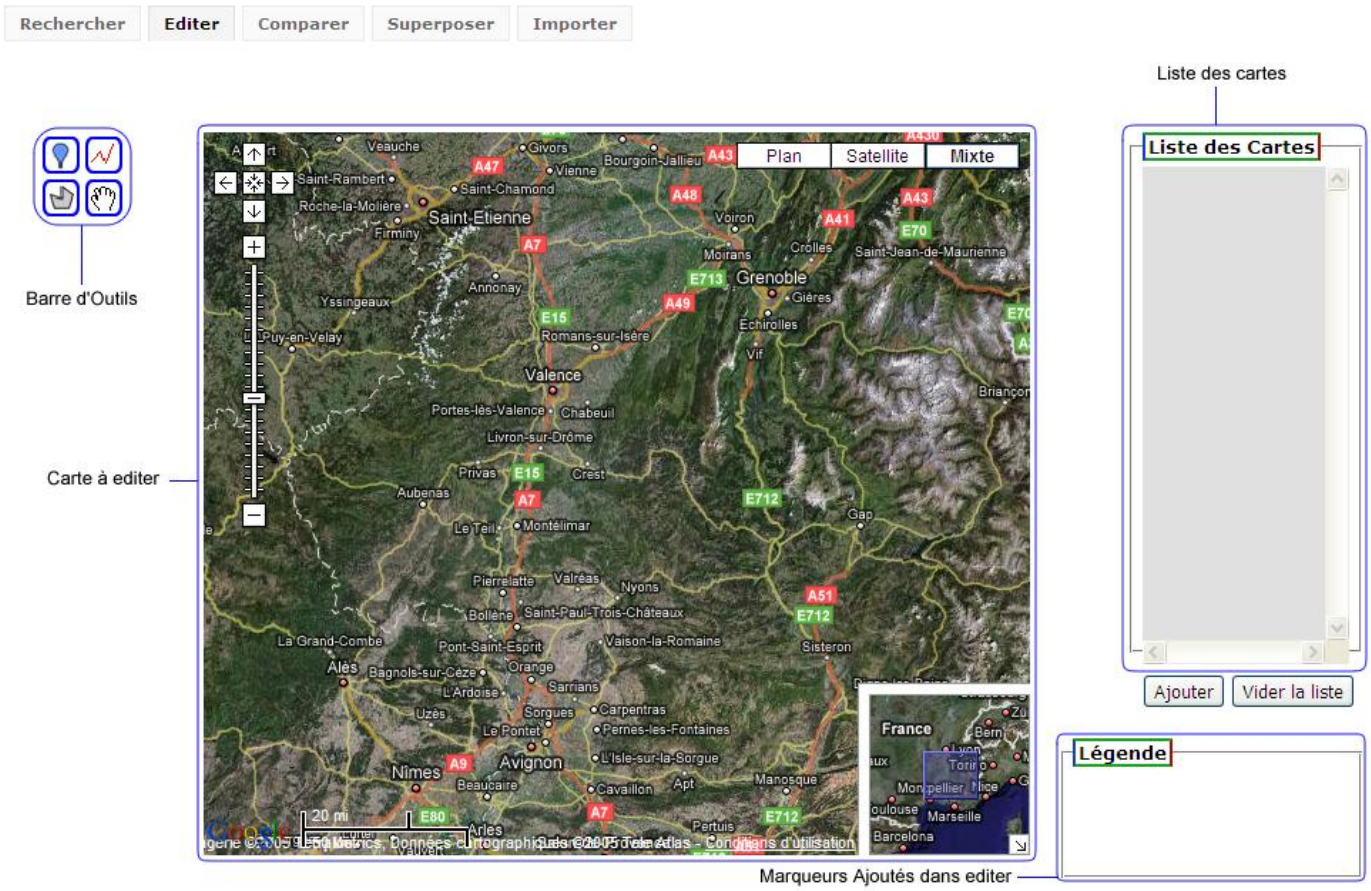


Figure 5.2.3.1 : interface de l'onglet Editer

L'interface de l'onglet éditer a en commun la liste des cartes qui est partagée avec tous les autres onglets sauf l'onglet «Importer », et aussi elle partage la légende avec l'onglet « Superposer », et elle se distingue des autres onglets par la barre d'outils.

5.3.1.2 Description de la Barre d'outils :

La Barre d'outils contiens quatre boutons correspondant chacun à une fonctionnalité précise :

5.3.1.2.1 Insérer des marqueurs ( ):

Cette fonction permet d'ajouter des marqueurs personnalisés avec un nom, une description et une icône différente de celle proposé par défaut par Google Maps, grâce à une fenêtre (Figure 5.2.3.2.1) permettant la saisie du nom, description et spécification du style propre au marqueur.

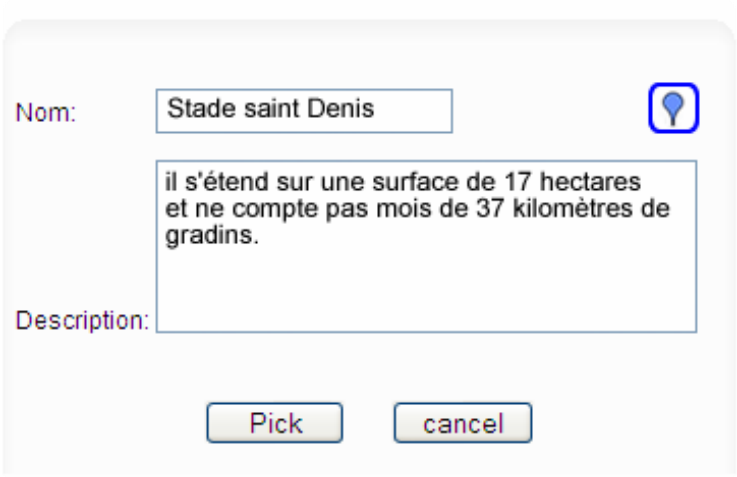


Figure 5.2.3.2.1 : fenêtre d'insertion des marqueurs

Après avoir saisie le nom et la description on peut ensuite attribuer un style au marqueur en cliquant sur le bouton ( ) qui fait apparaitre une fenêtre (Figure ci-dessous) proposant une multitude de style de marqueur aussi attrayant les uns que les autres.

Figure 5.2.3.2.1 : fenêtre des styles des marqueurs

Après sélection du style du marqueur, on revient à la première fenêtre pour valider tous les paramètres du marqueur et l'insérer dans la carte grâce à la fonction ajouterMarqueurs() décrite dans le listing suivant.

```
function ajouterMarqueurs(){
        var point = new GPoint(-92.27722, 34.74875);
        var marker = new GMarker(point);
        var eventmarqueursurcarte = GEvent.addListener(mapCourante[vueCourante], 'click',
        function(overlay, point) {
                if (point) {
                        var marque = createMarkerEd(point,description,nom_pin);
                        mapCourante[vueCourante].addOverlay(marque);
                         }
                         GEvent.removeListener(eventmarqueursurcarte); });}
```

Listing -ed1-

Lors de l'appel de ajouterMarqueurs() on ajoute un marqueur à la carte courante (map_ed) ce qui permet de l'afficher instantanément sur cette dernière et elle fait aussi appelle à createMarkerEd(point,description,nom_pin) qui elle l'ajoute à la structure de donnée partagé par le site .

5.3.1.2.1.1 Mise à jour de la légende :

Lors de l'ajout d'un marqueur dans une carte il s'ajoute automatiquement dans la légende si sont style n'y est pas déjà (Figure 5.2.3.2.1.1).

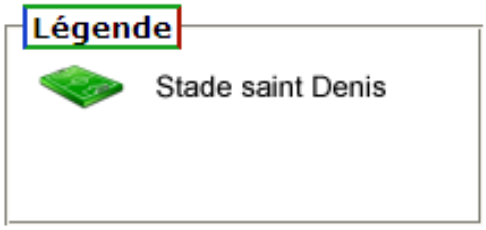


Figure: la légende des marqueurs.

5.3.1.2.2 Dessiner des polylines ( ) :

Cette fonction permet de dessiner des polyline ouvert et des polyline fermer pour mieux visualiser des résultats.

Pour insérer les polyline dans la carte, on fait appelle à la fonction DrawPolyLine() décrite dans le listing suivant.

```
function DrawPolyLine() {
        IconeLine();
        bool_l = true;
        fenetreDessin();
        newLine = new xccDrawLine(mapCourante[vueCourante],{
                iconPoints: blueIcon,
                iconGhostPoints: redIcon,
                lineColor: '#000066',
                lineWeight: 3,
                lineOpacity: 1
        });
}
```

5.3.1.2.2.1 Dessiner des polylines ouvertes :

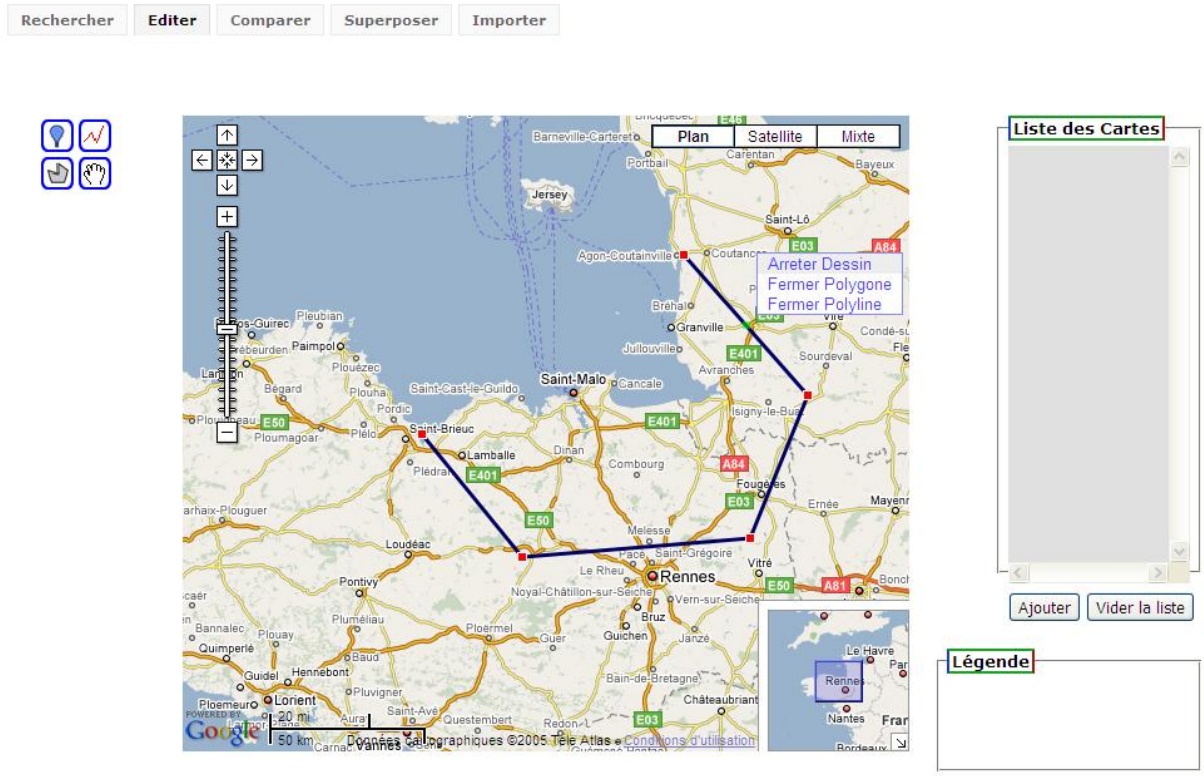


Figure: la légende des marqueurs.

Afin de garder une polyline ouvert faut exécuter la fonction stopDraw()(décrite dans le listing -5.2.3.2.2.1-) en faisant un click droit sur la carte et en sélectionnant « Arreter Dessin ».

```
function stopDraw(){
if(bool_l == true){newLine.stopDrawing(); bool_l=false; createLineEd("description","nomLine");}
if(bool_p == true){newPolygon.stopDrawing(); bool_p=false; createPolygoneEd("description","nomLine");}
        contextmenu.style.visibility="hidden";
        }
```

5.3.1.2.2.2 Dessiner des polyline fermes :

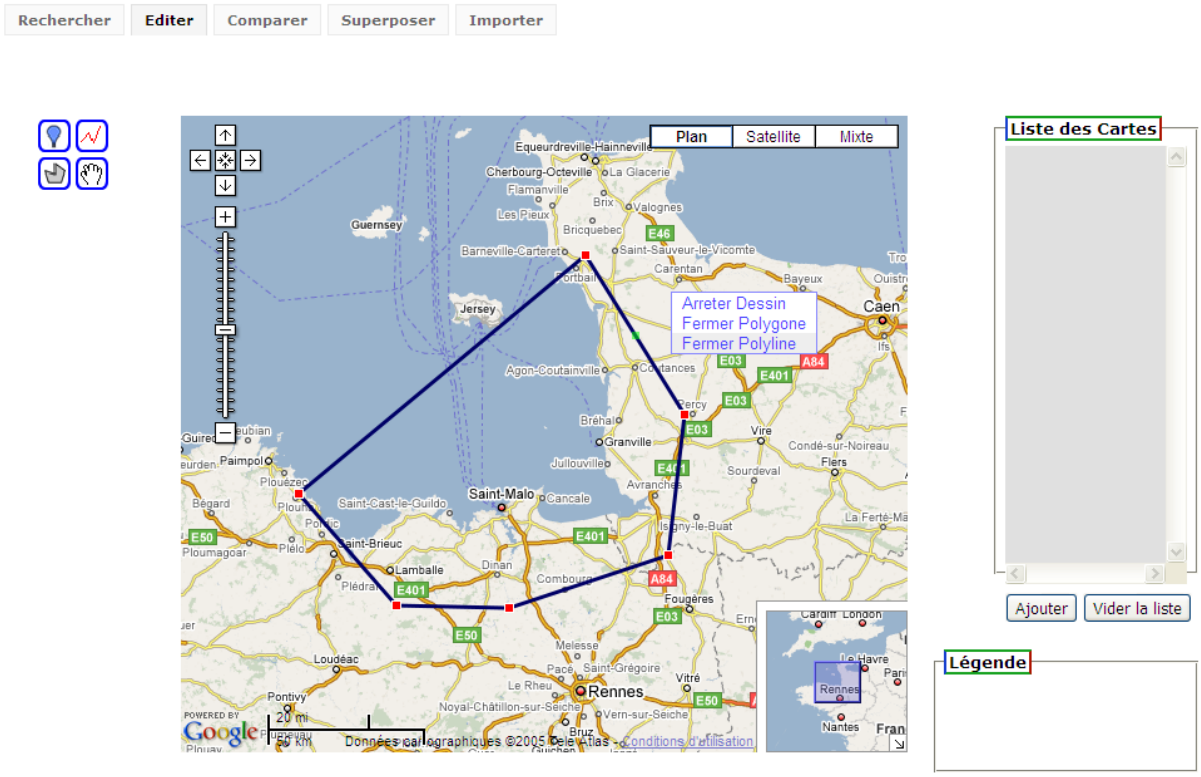


Figure : la légende des marqueurs.

Afin de fermer une polyline faut exécuter la fonction fermerPolyline() (décrite dans le listing -ed4-) en faisant un click droit sur la carte et en sélectionnant « Fermer Polyline».

```
function fermerPolyline(){
        if(newLine!= null && bool_l)
        if(newLine.myPoints.length > 2)
                {
                        newLine.polylines.push(new GPolyline([
                        newLine.myPoints[newLine.myPoints.length-1],
                        newLine.myPoints[0]
                        ], newLine.options.lineColor, newLine.options.lineWeight,
newLine.options.lineOpacity));
                        newLine.polylines[newLine.polylines.length - 1].myNumber =
newLine.polylines.length - 1;
        mapCourante[vueCourante].addOverlay(newLine.polylines[newLine.polylines.length - 1]);
                        //pour afficher un polyline fermer à partir de afficherDom
                        newLine.myPoints.push(newLine.myPoints[0]);
```

```
                    stopDraw();
                }else alert("vous ne pouvez pas fermer un polyline à une ligne!!!!!");
        else alert("ce n'est pas une Polyline!!");
}
```

5.3.1.2.3 Dessiner des polygone( ) :

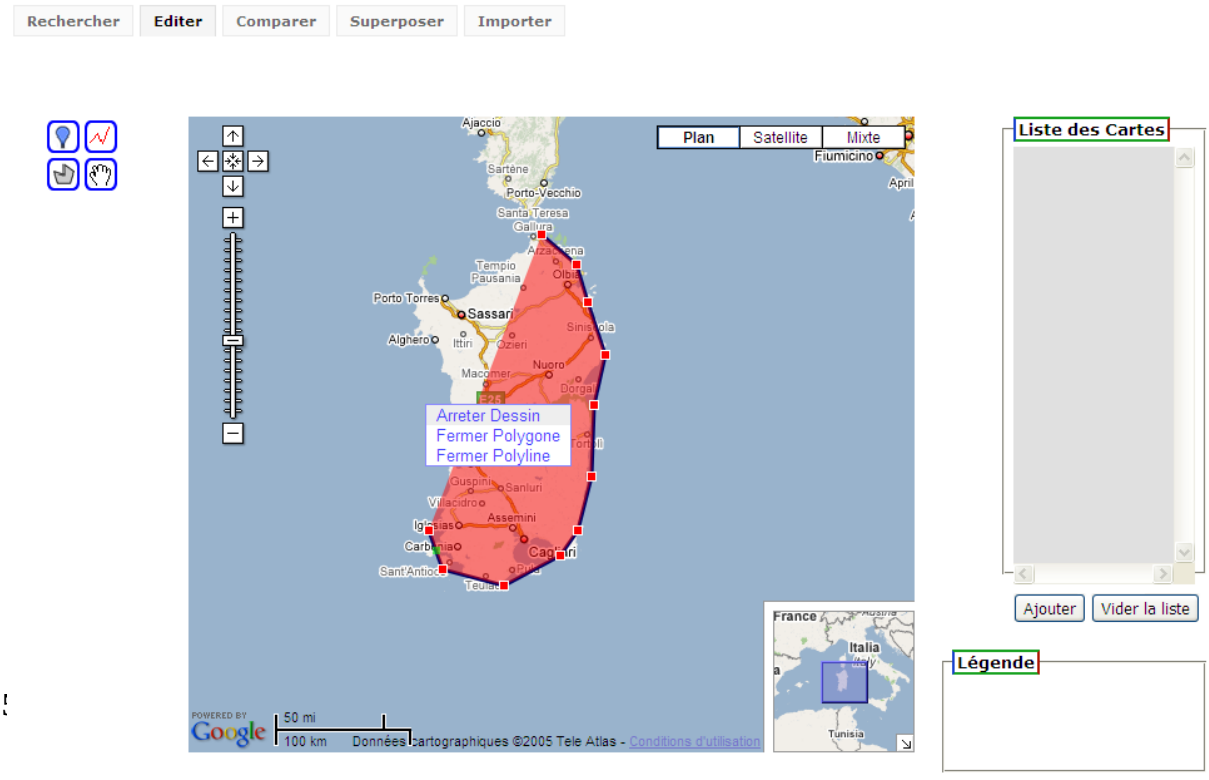
Cette fonction permet de dessiner des polygones ouverts et des polygones fermés pour mieux visualiser et cerner des résultats.

Pour insérer les polygones dans la carte, on fait appelle à la fonction DrawPolygone() décrite dans le listing suivant.

```
function DrawPolygon() {
    IconeLine();
    bool_p = true;
    fenetreDessin();
    newPolygon = new xccDrawPolygon(mapCourante[vueCourante],{
            iconPoints: blueIcon,
            iconGhostPoints: redIcon,
            lineColor: '#000066',
            lineWeight: 3,
            polygonColor: '#ff0000',
            polygonOpacity: 0.5,
            lineOpacity: 1
    });
}
```

5.3.1.2.3.1 Dessiner des polygones ouverts :

Afin de garder un polygone ouvert faut exécuter la fonction stopDraw()(décrite dans le Listing -5.2.3.2.3-) en faisant un click droit sur la carte et en sélectionnant « Arreter Dessin ».



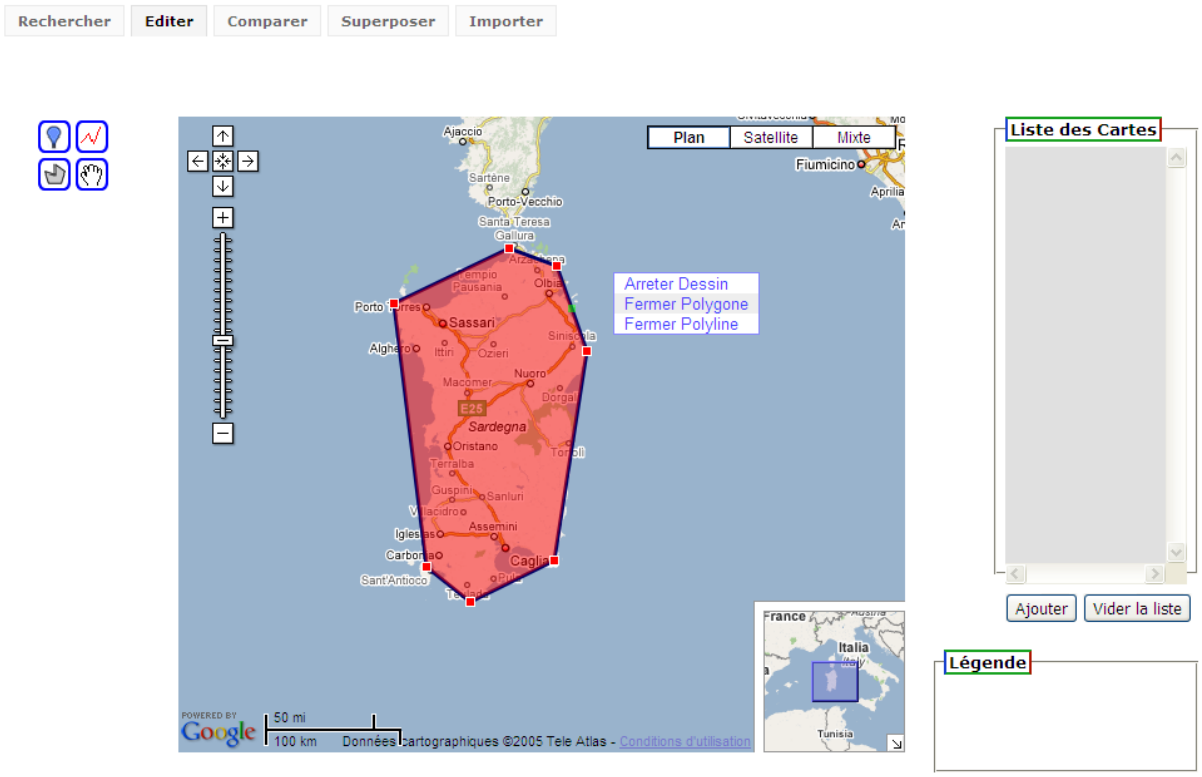


Figure : click droit pour fermer polygone

Afin de fermer un polygone faut exécuter la fonction fermerPolygone() (décrite dans le listing -ed6-) en faisant un click droit sur la carte et en sélectionnant « Fermer Polygone».

```
function fermerPolygone(){
        if(newPolygon!= null && bool_p)
                if(newPolygon.myPoints.length > 2)
                {
                        newPolygon.polylines.push(new GPolyline([
                        newPolygon.myPoints[newPolygon.myPoints.length-1],
                        newPolygon.myPoints[0]
                        ], newPolygon.options.lineColor, newPolygon.options.lineWeight,
newPolygon.options.lineOpacity));
                        newPolygon.polylines[newPolygon.polylines.length - 1].myNumber =
newPolygon.polylines.length - 1;
        mapCourante[vueCourante].addOverlay(newPolygon.polylines[newPolygon.polylines.length - 1]);
                        newPolygon.myPoints.push(newPolygon.myPoints[0]);
                        stopDraw();
                }else alert("vous ne pouvez pas fermer un polygone à une ligne!!!!!");
         else alert("ce n'est pas un Polygone!!");
}
```

5.3.2 Comparer:

La fonctionnalité de cet onglet consiste à comparer des cartes dans une interface qui possède deux zones de comparaisons, mais ces dernières peuvent dépasser deux si l'utilisateur veut comparer plus de cartes.

2.3.1 L'interface de l'onglet comparer :

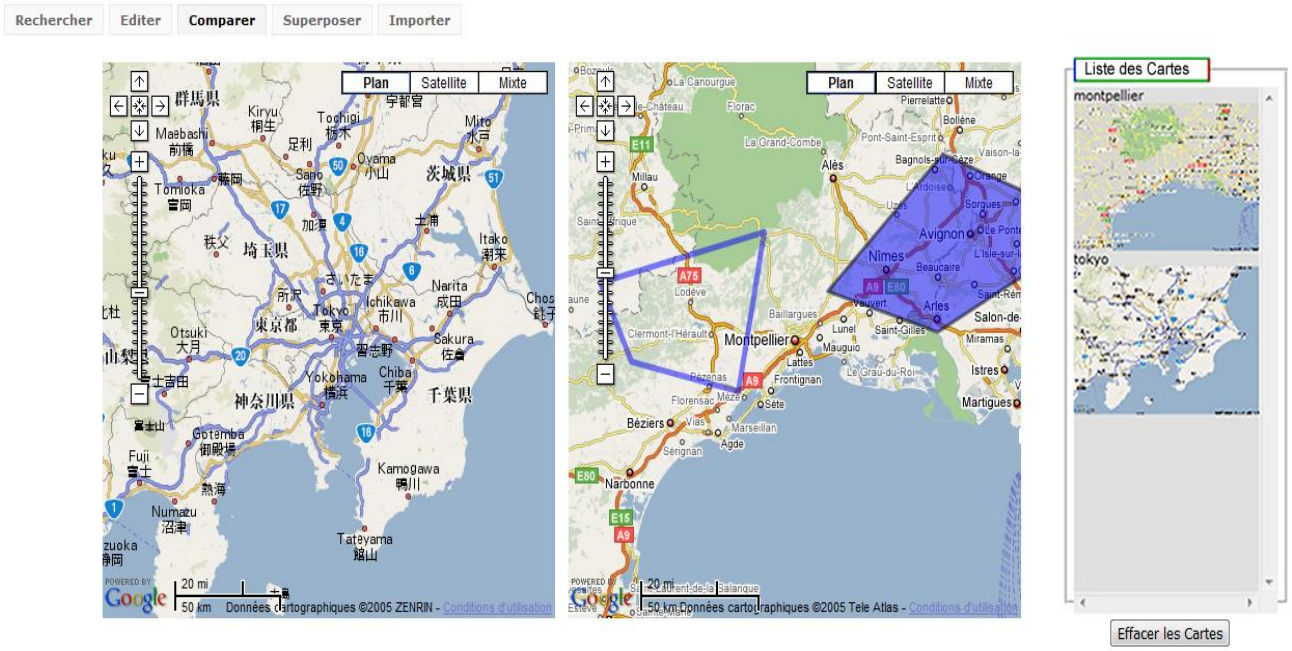


Figure 1 : interface avec deux zones de comparaison

L'utilisateur peut faire la comparaison des cartes, il pourra manipuler chaque carte en appliquant des zooms, des déplacements ainsi que des changements de type de cartes.

L'utilisateur peut comparer des cartes provenant des onglets : Rechercher, Editer, Superposer et des cartes importer du serveur ou bien du disque dur de l'utilisateur.

Pour effectuer une comparaison l'utilisateur doit choisir ses cartes à comparer dans la liste des cartes provenant des différents onglets. Par un simple click la carte s'affiche dans la première zone de comparaison et un second click sur une autre carte, la deuxième zone de comparaison reçoit la seconde carte. Voici le pseudo-code(décrit dans le Listing -co1-) qui a été implémenté pour cette fonction.

```
var DivTrouvé = "div_map_comparer1";
var tabBool = new Array();
```

```
tabBool["div_map_comparer1"] = 0;
tabBool["div_map_comparer2"] = 0;
tabBool["div_map_comparer3"] = 0;
tabBool["div_map_comparer4"] = 0;
var divcomparer = new Array ("div_map_comparer1",     "div_map_comparer2",
                             "div_map_comparer3", "div_map_comparer4");

Function Chercher_Div () {
        For (k=0; k<4; k++){
                If (tabBool[divcomparer[k]]==0) {
                        DivTrouvé = divcomparer[k];
                        tabBool[divcomparer[k]] =1;
                        booleen = 1;
                        break;
                        }
        }

}
```

Sur cette interface l'utilisateur ne peut comparer que deux cartes mais si le besoin de comparer plus de deux cartes se fait sentir, au troisième click l'interface se transforme en quatre zones de comparaisons et cette dernière est illustrée par la figure ci-dessous (Figure --).

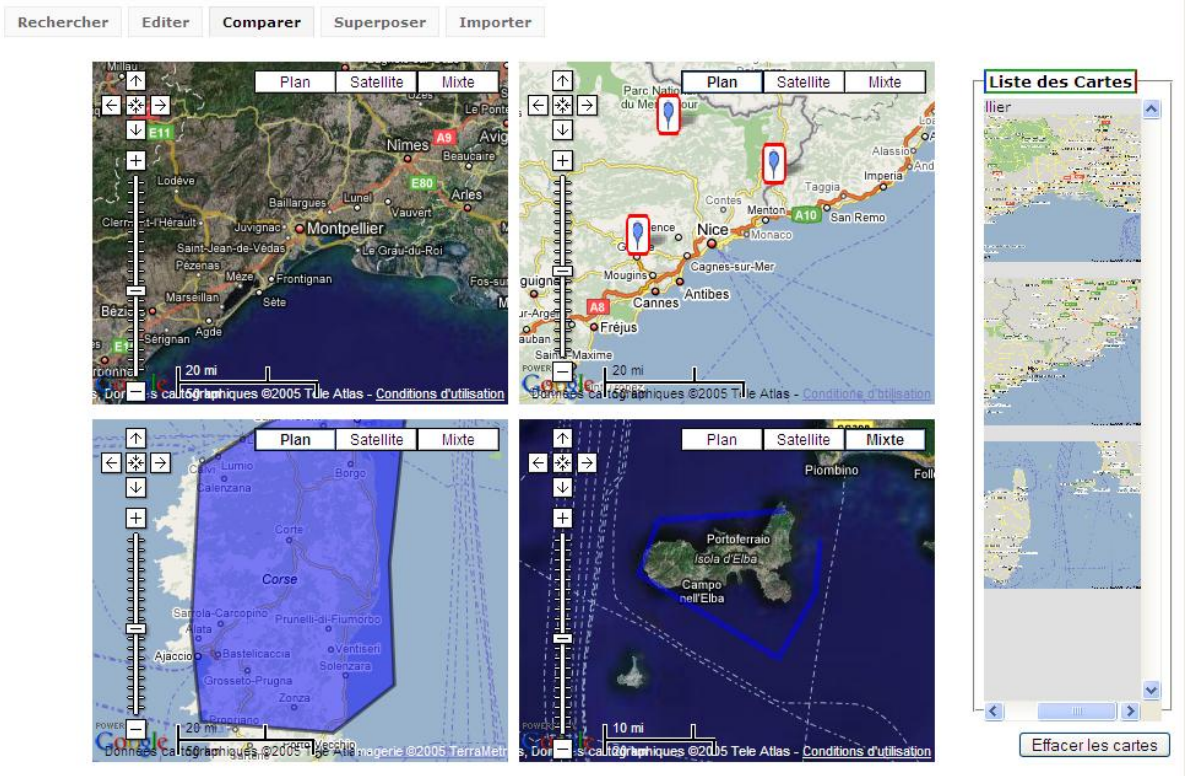


Figure 2 : <u>interface avec quatre zones de comparaison</u>

Cette interface ci-dessus permet à l'utilisateur de faire la comparaison par rapport à plusieurs cartes, la zone de comparaison numéro quatre peut prendre toujours la dernière carte dans la liste des cartes. Par contre il y a un bouton de commande Effacer les Cartes qui permet de supprimer toutes les cartes qui sont dans les différentes zones de comparaison. Cette interface ci-après illustre le début de la suppression des cartes dans les différentes zones de comparaison.

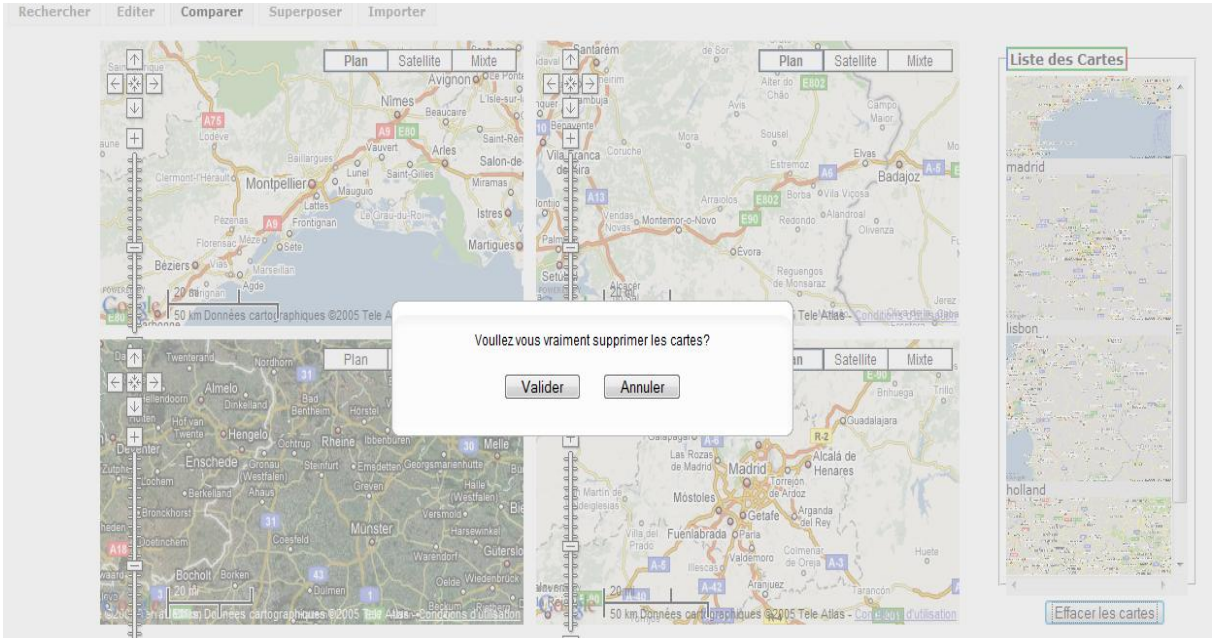


Figure 3 : interface pour effacer les cartes dans les zones de comparaison

Pour effacer complètement les cartes, l'utilisateur click sur le bouton valider et cette interface ci-après illustre nos propos.

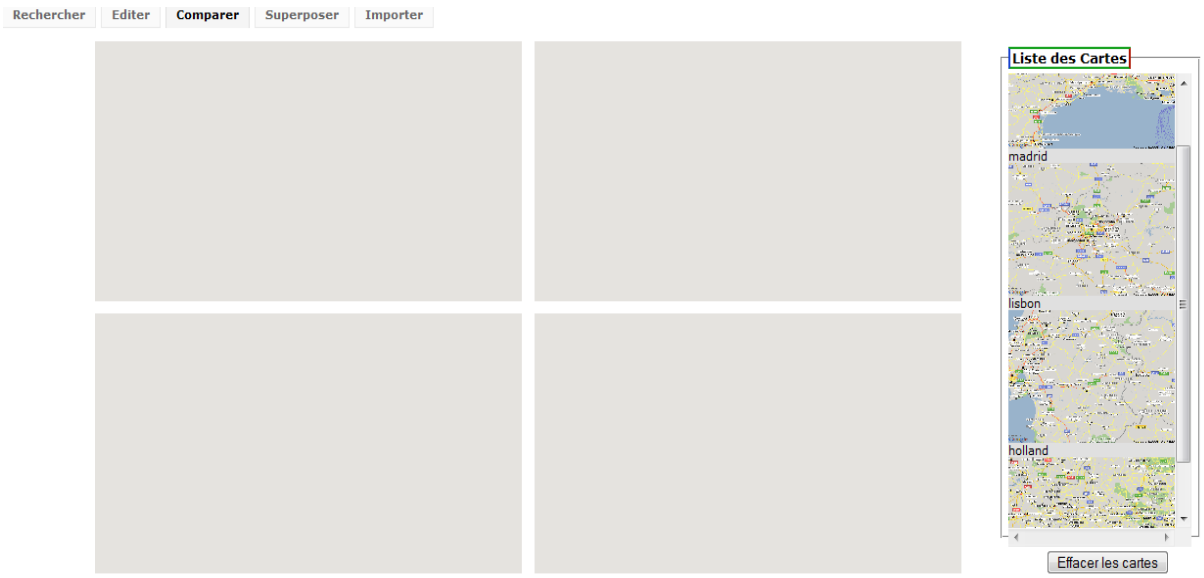
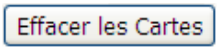


Figure 3 : interface sans les cartes dans les zones de comparaison

Voici ci-dessous le pseudo code du bouton de commande 


```
Function   Effacer_Carte () {
   For (k=0; k<4; k++) {
         document.getElementById (divcomparer[k]).innerHTML=" ";
         tabBool [divcomparer[k]]=0;
         }
}
```

5.3.3 Superposer

Lors de notre étude sur les différentes manière de représenter les résultats de carte géographique Google Maps, il nous est apparu que la superposition des résultats de 2 carte différentes pourrait faire l'objet d'une nouvelle fonctionnalité.

5.1 Fonctionnement

La superposition de plusieurs cartes ce fait en sélectionnant dans la liste des cartes les cartes que nous cherchons à fusionner.

Chaque carte sélectionné mets sur le plan de travail (carte géographique) les éléments de leur KML virtuel respectif.

La fonctionnalité superposer permet de faire la fusion des éléments contenu dans le plan de travail puis en créé un nouveau KML virtuel.

5.3.4 Importer

La fonctionnalité importer, permet d'intégrer dans notre géo-service des résultats de carte géographique contenue dans un fichier KML.

4.1 Fonctionnement

Cette fonctionnalité contient 2 mode pour importer un fichier KML:

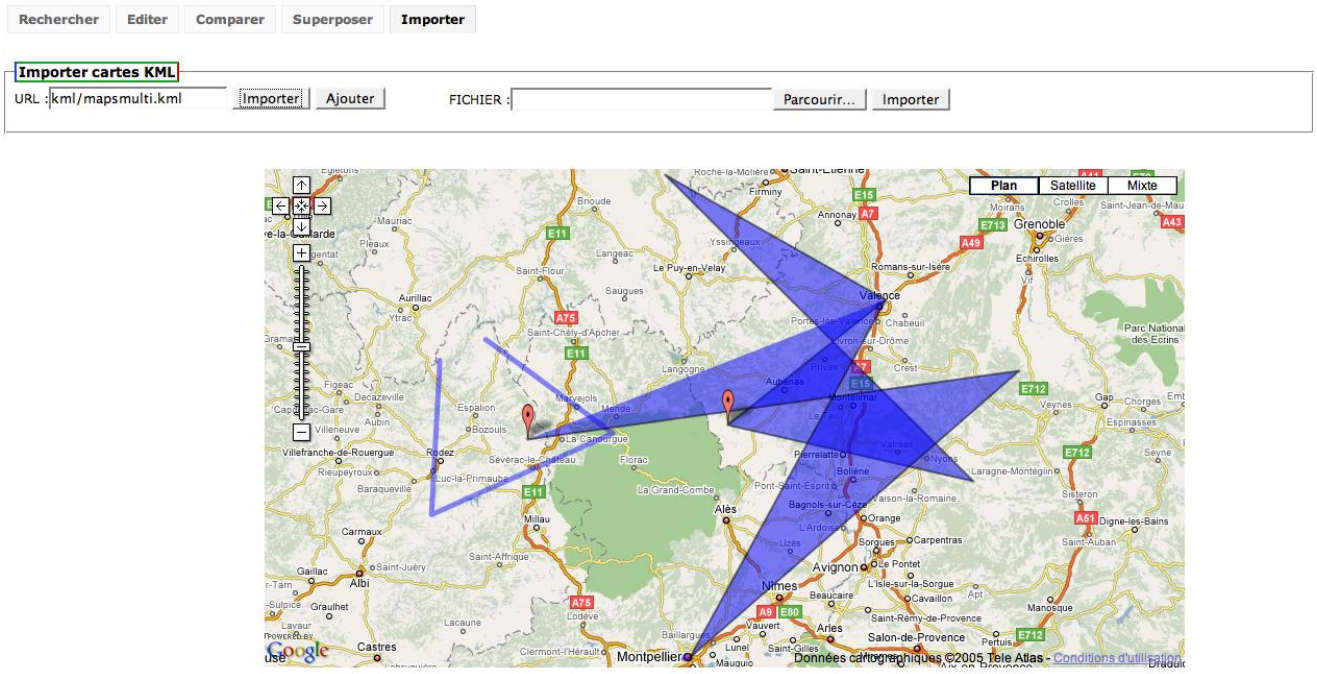
1.1 Url du fichier
1.2 Chemin Local du fichier

La méthode utiliser pour les 2 modes pour ouvrir un fichier est une instance de XMLHttpRequest (objet asynchrone d'AJAX).
Cette instance n'ouvre que des fichiers contenu sur le serveur où est hébergé le site.
Pour importer un fichier contenu sur le disque dur de l'utilisateur, nous devons dans un premier temps charger sur le site le fichier KML puis ouvrir avec XMLHttpRequest.

Chaque fichier KML importé est localisé sur la carte google maps avec à l'aide des méthode de la classe GLatLngBounds.

code:

```
function centre(markers) {
```




```
        var bounds = new GLatLngBounds();
    for(var i=0;i<markers.length;i++) {
                bounds.extend(markers[i]);
    }
        mapCourante[vueCourante].setZoom(mapCourante[vueCourante].getBoundsZoomLevel(bounds));
        mapCourante[vueCourante].setCenter(bounds.getCenter());
}
```

6.DISCUSSION

6.1. **Objectifs atteints**

L'application qu'on a faite, propose un géoservice qui améliore la navigation et la représentation des résultats de recherche sur une carte Google Maps, et c'est bien le but de notre TER. Et puisque il s'agit d'un travail exploratoire, on a étudié presque toute l'API Google Maps pour améliorer ce géoservice

6.2. Critiques et perspectives:

Notre application sera bien faite, si on avait un espace de travail, pour que les utilisateurs sauvegardent leurs propres cartes dans leurs espaces de travail.

Aussi si on pouvait exporter des fichier kml.

6.3 Compétences acquises

Au terme de notre projet, nous avons appris différentes et diverses compétences sur le domaine des services web mais aussi :

0- l'API Google Maps
1- JavaScript
2- Outil de versionning
3- PHP
4- Feuille de style CSS
5- Document Object Model

7.GLOSSAIRES

API (Application Programming Interface) : Elle permet de définir la manière dont un composant informatique peut communiquer avec un autre. C'est donc une interface de code source fournie par un système informatique ou une bibliothèque logicielle, en vue de répondre à des requêtes pour des services qu'un programme informatique pourrait lui faire.

DTD (Définition de Type de Document) : est une grammaire qui permet de vérifier la conformité du document XML, mais elle impose par contre le respect exact des règles de base de la norme XML.

DOM (Document Object Model) : est une recommandation du W3C qui décrit une interface indépendante de tout langage de programmation et de toute plate-forme, permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style de documents.

HTTP (HyperText Transfer Protocol) : est un protocole de communication client-serveur développé pour le World Wide Web. Le protocole HTTP peut fonctionner sur n'importe quelle connexion fiable, le serveur HTTP utilise alors par défaut le port 80.

MASHUP : est un site Web qui combine plusieurs applications Web au sein d'une application unique afin de créer une synergie et un service nouveau. Le contenu de ces applications peut provenir de sites web tiers et être mis à disposition par le biais d'API, c'est-à-dire des services Web (des programmes en ligne) autorisant l'extraction et le traitement d'informations

OASIS : est une organisation de standard dans le de eGouvernement et eBusiness, elle occupe la couche supérieure de standards c'est-à-dire la plus proche du niveau application, elle attire plus les partenaires industriels. Elle est sous la propriété d'Intel, plus sensible sur les intérêts commerciaux et moins stratégique.

W3C (World Wide Web Consortium) : est un organisme de normalisation fondé en octobre 1994 pour promouvoir la compatibilité des technologies du World Wide Web. Le W3C n'émet pas des normes au sens européen, mais des recommandations à valeur de standards industriels.

8.SUPPORT LOGISTIQUE ET WEB

SERVICE WEB

Dans le cadre du développement d'une application par plusieurs personnes, il est souvent nécessaire de se maintenir à jour sur les dernières modifications apportées à l'application et mettre a jour les différentes parties de celle-ci. Pour résoudre ce problème de communication et de partage de code source, nous avons utilise deux services web disponibles gratuitement : Google Code Hosting et Google Groupe de Discussion.

GOOGLE CODE HOSTING : Google Code est un site internet destine aux développeurs intéressés par le développement relatif a Google. L'entreprise y diffuse des codes sous licence libre ainsi que la liste des API utilisables. Project Hosting est un projet qui vise à apporter des services gratuits aux développeurs open-source de la même façon que le fait déjà Source Forge. Il est intégré au site Google Code.
Les projets qui peuvent y être soumis doivent être sous l'une de ses licences open source:
- Apache License 2.0
- Artistic License
- GNU General Public License 2.0
- GNU Lesser Public License
- MIT License
- Mozilla Public License 1.1
- New BSD License

SKYPE : est un logiciel propriétaire et service propriétaire de voix sur IP (VoIP) développé par les programmeurs pour les entrepreneurs de Kazaa et de The Venice Project. Nous avons utilisé skype version 3.6.0.248 pour la communication a distance, il nous a permi de faire des réunions de 1h 30 mn afin de régler certains problèmes et savoir chacun ou il en est pour le codage. Il nous permet aussi de faire la mise au point de l'avancement de notre projet.

Outils de développement

Afin de bien maintenir à chaque jour nos versions, la manière de se communiquer, le partage de nos fichiers, et la publication de notre travail sur internet nous avons utilise un certain nombre de logiciels qui seront énumères ci-après :

SmartSVN : Afin de maintenir a jour chacune de nos versions, nous avons utilises un logiciel client SmartSVN version 3.0.
SVN offre une interface graphique permettant de réaliser la plupart des taches qu'offre SVN en ligne de commande. L'explorateur de Windows s'enrichit des fonctionnalités suivantes:
- Superposition d'icones aux répertoires et fichiers permettant de visualiser instantanément l'état (a jour, modifie, en conflit...)
- Menu contextuel permettant de committer ou d'updater, a l'échelle d'un fichier, d'une sélection de fichiers ou encore d'un répertoire
- Possibilité d'ajouter en mode détails de l'explorateur des colonnes de type numéro de révision, état.

FireFox (avec plug-in firebug): Firefox est un logiciel multi-plateformes, compatible avec diverses versions de Microsoft Windows, Mac OS X et Linux. Le navigateur peut être personnalisé à

partir d'une base fiable et épurée. Plutôt que d'être fourni avec un nombre considérable d'options dans une distribution standard, Firefox accepte des centaines d'extensions et de thèmes graphiques, ce qui permet à chaque utilisateur de le modifier à son goût. Nous avons travaille avec la version 2.0.0.14

Easy PHP : est une plateforme de développement Web, permettant de faire fonctionner localement (sans se connecter à un serveur externe) des scripts PHP. EasyPHP n'est pas en soi un logiciel, mais un environnement comprenant deux serveurs (un serveur web Apache et un serveur de bases de données MySQL), un interpréteur de script (PHP), ainsi qu'une administration SQL PhpMyAdmin.
Par défaut le serveur Apache crée un nom de domaine virtuel (car local) http://127.0.0.1 ou http://localhost. La version 2.0 apporte les nouvelles versions des composants: Apache 2.2.3, PHP 5.2.0, MySQL 5.0.2, PHPMyAdmin 2.9.1.1, SQLiteManager 1.2.0.Par raison de système d'exploitation différent nous avons utilisé WAMP, XAMP, MAMP.

HERBERGEUR FREE: est une entité ayant pour vocation de mettre à disposition des internautes des sites web conçus et gérés par des tiers. Il donne ainsi accès à tous les internautes au contenu déposé dans leurs comptes par les webmestres souvent via un logiciel FTP ou un gestionnaire de fichiers. C'est l'hébergeur Free que nous avons utilisé pour mettre notre site en ligne dont l'url est **http://univers.seed.free.fr/**