Universitat Oberta de Catalunya
www.uoc.edu

Treball final de Carrera

Desenvolupament d'edició geogràfica segons l'estàndard WFS-T sobre l'API de Google Maps

Alumne: Francisco Ruché Zarza
Consultor: Albert Botella Plana

Enginyeria Tècnica de Telecomunicacions, especialitat Telemàtica
Semestre setembre 2008 – febrer 2009

RESUM

Els Sistemes d'Informació geogràfica, en endavant SIG, van aparèixer als anys seixanta per intentar relacionar dades purament geogràfica amb altres tipus d'informació, amb l'objectiu d'extreure coneixement que d'altra forma seria inviable.

Aquest sistemes, sempre han estat unes eines d'ús restringit, però amb l'aparició d'Internet, tant el programari, com les dades disponibles s'han posat a l'abast d'un públic molt més nombrós. L'últim capítol d'aquesta expansió és deguda a l'aparició d'iniciatives com les de *Google Maps*, permetent gaudir d'un programari i unes cartografies d'accés pràcticament universal.

El present treball tractarà d'aprofitar els serveis proporcionats per *Google Maps* i les eines lliures disponibles, per construir unes llibreries *Javascript* d'ús obert que permetin afegir serveis de persistència als serveis de cartografia ja disponibles.

S'iniciarà el treball amb una introducció teòrica als sistemes GIS i en especial a les tecnologies necessàries per a poder desenvolupar un sistema SIG sobre Web, els elements de programari, la seva configuració i els estàndards a seguir.

Es farà un disseny basat en el patró arquitectònic Model-Vista-Controlador a través d'una aproximació basada en objectes, per construir totes les utilitats necessàries d'un programari de gestió persistent d'objectes geomètrics bàsics geo localitzats. En especial, es farà un esforç en el disseny de la interfície gràfica d'usuari, per tal de donar el major nombre possible de capacitats d'ús i comoditat a l'hora de fer servir el programari.

ÍNDEX DE CONTINGUTS

ÍNDEX DE FIGURES

ÍNDEX DE TAULES

1. PLANIFICACIÓ I SEGUIMENT DEL PROJECTE

Tot seguit, descriurem els objectius del projecte, el pla de treball a seguir amb les seves tasques i fites principals així com la distribució temporal d'aquestes.

També es farà un anàlisis dels riscos que poden aparèixer durant l'execució del projecte i les seves possibles solucions, el material necessari per construir el producte final, i per últim, un seguiment de cadascuna de les fites.

1.1. JUSTIFICACIÓ DEL TFC

L'objectiu final del TFC, és el d'oferir a la comunitat un seguit de components *Javascript*, que permetin dotar de persistència a l'edició d'elements geogràfics, tot aprofitant la cartografia global i els serveis donats per l'API de *Google Maps*.
La funcionalitat d'aquests components, consistirà a gestionar els següents elements geogràfics:

- Elements puntuals
- Elements lineals
- Elements d'àrea simple

Per tal d'aconseguir aquest objectiu principal, caldrà integrar i configurar, diferents productes de programari lliure, per tal de construir el sistema complert
Opcionalment, es podran construir diversos exemples d'utilització dels components dissenyats, que puguin servir de demostració d'utilitat del sistema.

1.2. OBJECTIUS

Per tal d'assolir l'objectiu principal, o producte final, caldrà plantejar els següents objectius:

- Adquirir els coneixements genèrics necessaris sobre sistemes SIG, per a poder desenvolupar el TFC.
- Conèixer alguns dels productes de programari lliure SIG més utilitzats, com ara *Geoserver* **[9]** o *PostGIS* **[8]**.
- Estudiar els conceptes i tècniques bàsiques utilitzades pels servidors d'informació geogràfica via web per a oferir serveis als clients; conceptes com XML, serveis web,...
- Aprendre a configurar programari, no específicament SIG, que formant part del sistema, ofereixen les capes de persistència (servidor de BBDD), i presentació (servidor http).
- Conèixer els diferents estàndards oferts pe OGC **[5]** (*Open Geospatial Consortium*), que es faran servir en el desenvolupament del projecte, estàndards com:
  - WMS (*Web Map Service*).
  - WFS-T (*Transactional Web Feature Service*)
  - GML (*Geography Markup Language*).
- Entendre l'API de *Google Maps*, per a poder construir els components necessaris, que aprofitin les seves funcionalitats.

- Aprendre *Javascript*, i la seva forma de construir components que implementin les funcionalitats requerides.
- Integrar diferents components ja existents, per a poder crear un sistema d'edició geogràfica i oferir-lo com a serveis a la comunitat.

1.3. PLA DE TREBALL

El pla de treball, considera la relació de les tasques, la seva distribució en el temps i la seva duració, com la part més important de la planificació d'un projecte. A continuació farem una exposició de les decisions preses en aquests aspectes.

1.3.1. Relació de tasques

Un dels problemes més grans trobats, és la planificació del desconegut, ja que no es tenen els coneixements tècnics necessaris per avaluar les durades ni els riscos que poden sorgir. Per tant, és molt important, fer-se abans de la planificació, una imatge de conjunt de la totalitat del sistema a instal·lar, fent les proves necessàries per a poder avaluar els temps que faran falta a les diverses fases del desenvolupament.

Un altre dels aspectes més destacables d'aquest projecte, és la necessitat d'adquisició de coneixements, examinant i estudiant una gran quantitat de documentació, cosa per la qual, s'ha donat prou rellevància a les hores assignades a aquestes tasques.

El mètode de treball adoptat, es dividirà en 4 fases:

- **Fase 1**: Cerca d'informació
- **Fase 2**: Instal·lació del sistema SIG
- **Fase 3**: Desenvolupament dels components, prova i exemple d'ús

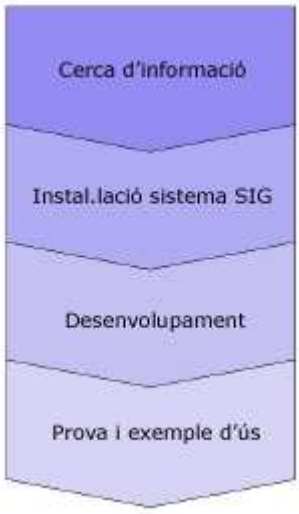


Figura 1.1 Mètode de treball

Cal destacar, que si bé, la definició del pla de treball i la generació de la documentació final, són tasques a realitzar, no s'ha considerat fases fonamentals del mètode de treball, ja que, la primera, és imprescindible, i la segona és el resultat de les anteriors.

S'ha estimat un horari de treball, sempre de dilluns a divendres, de 2 hores diàries, i de 3 hores a la fase crítica del desenvolupament i prova dels components a dissenyar.

Tenint en compte les decisions anterior, s'han planificat les següents tasques, amb la seva càrrega estimada de treball, en hores:

Tasques i sub tasques		***hores***
Definició del pla de treball.		27
	Estudi previ de viabilitat.	21
	Redacció del pla de treball.	6
Fase 1- Cerca i estudi de documentació.		20
	Cerca d'informació genèrica sobre SIG.	10
	Cerca i estudi d'aspectes tecnològics involucrats.	10
Fase 2 – Instal·lació i prova del sistema		27
	Cerca d'informació sobre els elements de programari necessaris.	6
	Instal·lació, configuració i prova del programari.	12
	Redacció de documentació sobre instal·lació i prova.	6
	Revisió de l'esborrany.	3
Fase 3 – Desenvolupament i prova dels components Javascript.		80
	Implementació d'un component AJAX per sobre de l'API *Google Maps*.	9
	Implementació de la gestió de punts.	30
	Implementació de la gestió de línies.	6
	Implementació de la gestió d'àrees simples.	6
	Construcció d'un exemple basat en els components desenvolupats.	12
	Redacció de la PAC3.	12
	Revisió de l'esborrany.	5
Generació de documentació final.		23
	Redacció de la memòria.	6
	Construcció de la presentació.	12
	Revisió de l'esborrany.	5

Taula 1.1 Relació de tasques

1.3.2. Fites principals

Com a fites principals, s'han escollit, les dates d'entrega oficials de les diferents PAC.
Existeixen unes dobles fites associades a l'entrega de l'esborrany i l'entrega definitiva, de cada PAC. Aquestes dos fites, s'han separat 3 dies laborables, per tal de donar temps, encara que sigui mínim, per la correcció del consultor i la posterior modificació del document.

Fita principal	***Lliurament esborrany***	***Lliurament definitiu***
F1-Pla de Treball	-	10 de Setembre
F2-PAC2	30 d'Octubre	4 de Novembre
F3-PAC3	28 d'Octubre	9 de Desembre
F4-Memòria final	28 de Desembre	5 de Gener

Taula 1.2 Fites i lliuraments

Els continguts que s'han decidit presentar a cadascuna de les entregues, són els següents:

Fita principal	***Continguts***
F1-Pla de Treball	Definició del pla de treball
F2-PAC2 Fase 1 i 2	1 – Marc teòric 2 – Planificació del projecte i seguiment (Actualització) 3 – Desenvolupament del treball i productes obtinguts 3.1 – Descripció global del sistema implementat 3.2 – Instal·lació i configuració del programari
F3-PAC3 Fase 3	1 – Marc teòric 2 – Planificació del projecte i seguiment (Actualització) 3 – Desenvolupament del treball i productes obtinguts (Descripció de classes implementades) Maqueta del treball pràctic, i exemple de funcionament
F4-Memòria final	Memòria complerta Presentació Treball pràctic

Taula 1.3 Fites i continguts

1.3.3. Planificació

Una vegada definides les tasques, amb la seva càrrega de treball, i la seva precedència, passem a distribuir-les en el temps, mitjançant diagrames de Gantt.
S'han tingut en compte només els dies feiners, desestimant les festes locals i estatals com a dies de treball, encara que es podien aprofitar per a avançar feina.

Als diagrames adjunts, s'han diferenciat els tipus de tasques per colors:

- **Gris**: tasques de cerca de documentació.
- **Verd**: tasques de redacció i revisió dels lliuraments.
- **Blau**: tasques d'instal·lació i implementació.

Les fites s'assenyalen amb un rombe negre, i els lliuraments oficials amb un rombe vermell en el cas de que no coincideixin amb una fita pròpia.

Tasca	Durada
Definició del pla de Treball	**9 días**
Estudi prèvi de viabilitat	7 días
Redacció del pla de treball	2 días
Lliurament pla de treball	0 días
Cerca i estudi de documentació	**10 días**
Cerca i estudi d'aspectes tecnològis involucrats.	5 días
Cerca i estudi dels estàndards OGC necessaris.	5 días
Insta.lació i prova del sistema	**15 días**
Cerca d'informació sobre els elements de programari necessaris.	3 días
Instal.lació, configuració i prova del programari.	6 días
Redacció de documentació sobre instal.lació i prova	3 días
Entrega esborrany PAC2	0 días
Modificació esborrany	3 días
Entrega PAC2	0 días

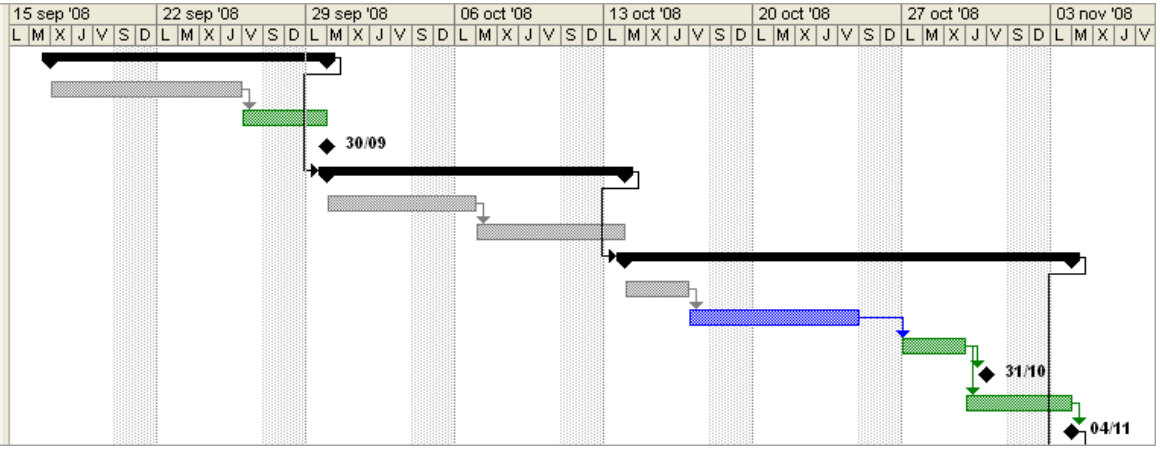


Figura 1.2 Diagrama de Gantt fases 1 i 2

Tasca	Durada
Desenvolupament i prova dels components Javascript.	**21 días**
Implementació d'un component AJAX per sobre de l'API Google-Maps	3 días
Implementació de la gestió de punts	5 días
Implementació de la gestió de linies	2 días
Implementació de la gestió d'àrees simples	2 días
Construcció d'un exemple basat en els components desenvolupats	4 días
Redacció PAC3	2 días
Entrega esborrany PAC3	0 días
Modificació esborrany	3 días
Entrega PAC3	0 días
Entrega límit PAC3	0 días

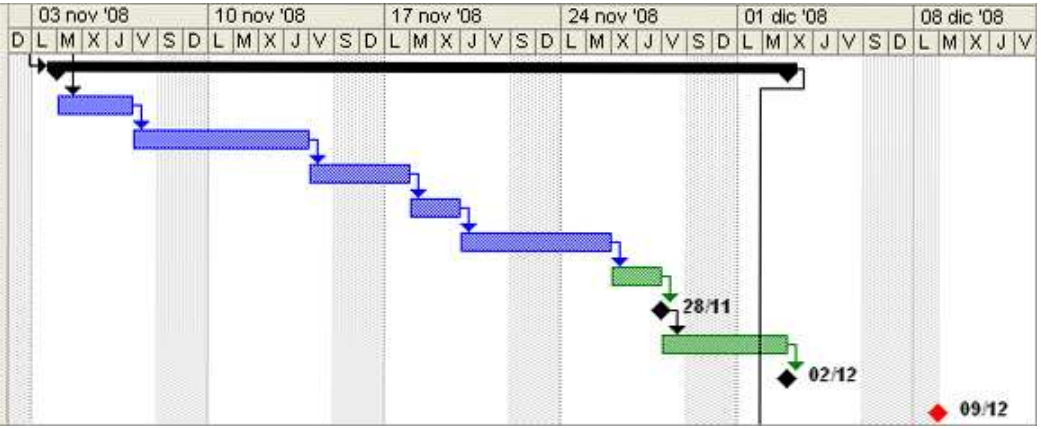


Figura 1.3 Diagrama de Gantt fase 3

Tasca	Durada
Generació de documentació final	**11 días**
Redacció de la memòria.	3 días
Construcció de la presentació.	5 días
Entrega esborrany memòria final	0 días
Modificació esborrany	3 días
Entrega memòria final	0 días
Entrega límit entrega memòria final	0 días

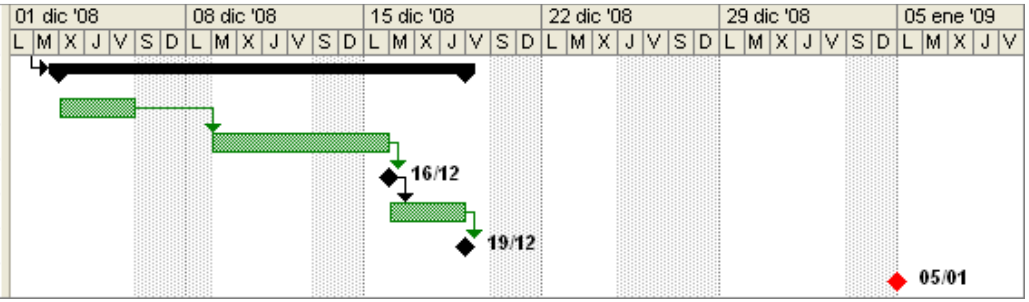


Figura 1.4 Diagrama de Gantt fase 4

1.3.4. Avaluació de riscos

Tenint en compte la situació personal de l'alumne, la disponibilitat d'horaris degut a la feina, família, i compromisos amb altres assignatures, s'han pogut identificar els següents riscos, als que donem una probabilitat que apareguin, i un nivell d'impacte sobre el treball global:

Riscos identificats	***Probabilitat***	***Impacte***
Malaltia comuna	Alta	Baix
És molt probable, que al llarg del semestre, e un moment o d'altre, es pugui patir una malaltia comuna, que podria fer retardar el treball en un parell de dies. Davant d'aquest risc, només cal portar el treball al dia, i recuperar les hores perdudes amb un ritme de treball més alt.		
Avaria de maquinari	Baixa	Baix
Encara que no sigui probable, s'ha pres la decisió de disposar de dos ordinadors configurats de la mateixa forma, per tal de poder canviar d'un a l'altre sense pèrdua de temps en la reparació, i fent còpies diàries en disc dur extern de les informacions de treball.		
Problemes de configuració	Alta	Alt
Al llarg de la Instal·lació del sistema SIG, poden aparèixer dificultats de configuració o funcionament, que s'hauran de resoldre el més aviat possible. Si aquest fos el cas, s'hauria de demanar ajuda, als serveis de suport disponibles, dels fabricants del programari en qüestió, o a grups d'usuaris.		
Compromisos amb altres assignatures	Alta	Alt
Degut a la gran quantitat d'assignatures que aquest semestre es cursaran, no s'ha pogut planificar el calendari per evitar coincidències en els lliurament d'activitats. Malgrat tot, s'ha pres la decisió de prioritzar aquest TFC, davant d'altres compromisos.		

Taula 1.4 Resum dels riscos identificats

1.4. MATERIAL

Passem a detallar tot el material necessari per a realitzar el TFC, que s'ha dividit en material de maquinari i comunicacions, i en material de programari.
Els costos derivats d'aquest material, és nul, ja que, o bé se'n disposa en el cas del maquinari, o bé és lliure en el cas del programari.

1.4.1. Maquinari i comunicacions

Es disposa dels següents materials:

- Dos ordinadors personals compatibles en xarxa, amb el programari estàndard recomanat per la UOC.
- Connexió ADSL, 4 Mbps,

Per tal d'evitar les pèrdues de temps en cas de possibles fallades del sistema operatiu, s'ha instal·lat en tots dos ordinadors disponibles, el mateix sistema operatiu, les mateixes eines, així com el programari específic necessari per a la constitució del sistema SIG.

1.4.2. Programari

Es necessitarà el següent programari:

- Programari ofimàtic genèric
  - ***Microsoft Office Visio 2003***: Aquest programari, ofereix la possibilitat de construir esquemes i gràfics de gran qualitat per a poder il·lustrar certes parts del projecte.
  - ***Microsoft Office Project 2003***: Programari que permet la gestió i planificació del projecte.
- Programari específic SIG
  - ***Java SE 6***: Màquina virtual i eines de desenvolupament Java, necessàries per a poder fer servir altres elements de programari com Geoserver o Tomcat **[6]**.
  - ***Servidor Tomcat 6.0***: Servidor HTTP i contenidor J2EE que contindrà part del sistema.
  - ***Geoserver 1.6.5***: Servidor SIG en la seva versió J2EE que permet incorporar els estàndards WFS-T i GML.
  - ***PostgreSQL* [7]**: Base de dades de caràcter genèric que serà aprofitada per donar persistència a les dades geogràfiques.
  - ***PostGIS 1.3.3***: Extensió per a PsotgresSQL que aporta les funcionalitats de dades necessàries per a poder fer servir dades específiques OGC.
  - ***Aptana Studio 1.1***: Eina de desenvolupament de JavaScript, i d'altres elements.
- Programari d'edició de vídeo
  - ***Camtasia***: Eina per editar i captura vídeo directament de la pantalla de l'ordinador.

1.5. SEGUIMENT DEL PROJECTE

Fita 2, 4 de Novembre de 2008

Fins al moment actual, el projecte es segueix de la forma planificada, sense cap entrebanc significatiu. S'ha completat la fase de cerca d'informació, la instal·lació i prova del sistema, així com les primeres proves de desenvolupament dels components Javascript.

L'únic contratemps ha estat la manca de temps per a aportar la part de documentació planificada que feia referència a la instal·lació del sistema. També cal remarcar la dificultat d'encabir molta informació en un espai tant reduït, cosa per la qual s'han tingut que resumir molts conceptes que s'haurien d'explicar més àmpliament. En concret s'està considerant la possibilitat de modificar la part referent al marc teòric per poder disposar de més espai per la documentació de la implementació.

Fita 3, 9 de Desembre de 2008

S'han produït importants canvis en el desenvolupament del projecte que han causat un reajustament de les dates d'entrega previstes, passant a ser les dates oficials d'entrega, és a dir s'han retardat les dates d'entrega de la fase 3 i 4.

El motiu principal d'aquest canvi del pla de treball, ha estat un profund desconeixement de les tecnologies involucrades en el treball, i la voluntat d'adaptar-se contínuament als avenços aconseguit durant el procés d'aprenentatge.
S'ha considerat un bon motiu pel retard del treball evolucionar cap a enfocaments més moderns que els proposats des d'un inici. En concret s'han variat els següents trets:

- Passar d'un model de programació procedimental a un orientat a objectes, amb la dificultat d'aconseguir-ho amb un llenguatge no dissenyat per a tal paradigma de programació com Javascript, però d'altra banda ineludible.
- Incorporar patrons de disseny, que no es tenien pensats en un principi, però que s'han vist molt útils a l'hora de la implementació final, patrons com el Model-Vista-Controlador, o el patró Observable.
- Utilitzar unes llibreries molt complexes per a proporcionar una interfície gràfica d'usuari el més usable possible. Encara que s'hagi tingut que començar des de zero, el resultat es creu que val l'esforç.

El retard ha estat d'una setmana, però ha estat suficient com per fer un canvi tan radical a aquestes alçades del projecte amb uns resultats que són molt satisfactoris.

Realment amb els canvis introduïts, s'ha comprovat que molts del problemes que es tenien per falta d'estructura en el disseny, s'han anat solucionant d'una forma molt adequada, encara que falten molts aspectes per arrodonir, es creu que s'està arribant als objectius marcats.

En aquesta entrega de la fase 3, s'incorpora un model del treball pràctic, que permet l'edició

de punts sobre el mapa i també un esbós de la presentació per tal de començar a provar les eines que es faran servir per a la entrega final.

Fita 4, 5 de Gener de 2009

S'han aconseguit tots els objectius planificats, és a dir l'edició de tots els tipus d'elements geomètrics considerats (punts, línies i àrees), dintre de les llibreries *Javascript* dissenyades.

Podem destacar la variació al codi font que s'ha hagut de fer per tal d'adaptar-se a nous patrons de disseny que s'han inclòs en aquesta darrera fase, així com l'increment d'hores dedicades a fer aquests canvis.

Cal afegir la gran dificultat que s'ha tingut a l'hora de fer la presentació en vídeo del projecte, doncs no es tenia cap mena d'experiència amb les eines de captura i edició, cosa per la qual, s'ha invertit mol temps en adquirir-les.

També s'han tingut dificultats amb el maquinari disponible, ja que no suportava la gran càrrega de procés requerit. Per solucionar aquest problema, s'ha hagut d'adquirir un maquinari nou amb els requisits necessaris, tant de tarja de vídeo com de memòria.

2. MARC TEÒRIC

En aquest capítol s'ofereixen les bases teòriques necessàries que es faran servir al llarg del desenvolupament del treball. Es comença amb una petita introducció al SIG, la seva evolució històrica, una descripció dels elements que el constitueixen i per últim una explicació del tipus de dades que manipula un SIG.

Es continuarà amb una explicació sobre els sistemes SIG sobre web, amb el focus posat en els serveis proporcionats per *Google*, ja sigui *Google Maps* com *Google Earth* i les seves API com a pas fonamental per a construir un *Mashup*, objectiu principal del treball.

Per últim s'explicaran les principals tecnologies que es faran servir per desenvolupar el producte final, tecnologies com XML, Javascript, AJAX i els estàndards OGC, en especial WFS-T.

2.1. Introducció als SIG

Podríem definir un SIG, com el conjunt de recursos que permeten la gestió d'informació geogràfica.

Encara que sigui una definició molt poc precisa, és el suficientment oberta perquè pugui emparar molts conceptes, tecnologies i metodologies, que avui en dia no paren d'evolucionar.

L'objectiu principal d'un SIG, no és oferir una representació gràfica dels sistemes que modelen, només és una de les seves possibilitats. Si s'hagués d'escollir l'objectiu més rellevant, seria el d'oferir coneixement d'una realitat geogràfica a través de diversos procediments, és a dir, donar la possibilitat d'interrogar al sistema SIG per tal d'obtenir respostes concretes, com per exemple preguntes del tipus: Quina zona de la vall es podria inundar si es desbordés el riu?, Quantes botigues i a quines distàncies es troben d'un punt determinat?, o bé, Quina és la ruta per carretera més curta entre dos poblacions?.

Per a poder respondre a aquestes preguntes, els sistemes SIG, han de poder representar realitats de situació física de molts diferents elements, ja siguin elements topogràfics del terreny, elements de situació d'objectes diversos, o elements amb atributs específics.

Aquesta representació es fa per capes temàtiques, cadascuna de les quals, ofereix informacions sobre un aspecte concret, i a través del SIG, es poden relacionar per tal d'oferir informació i respostes complexes.

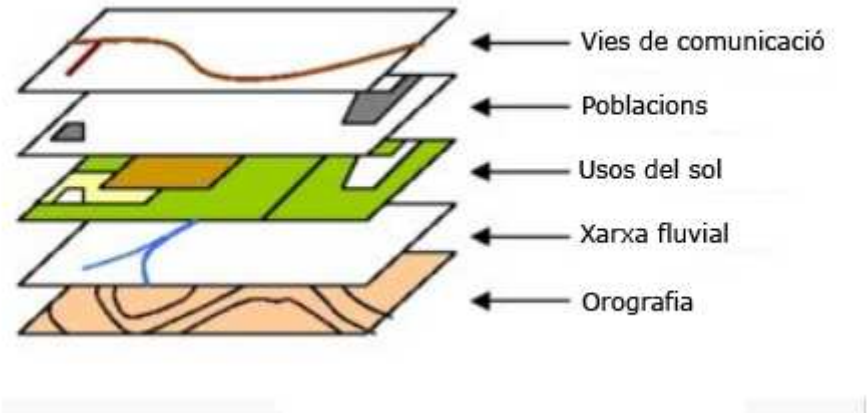


Figura 2.1 Capes d'un sistema SIG

2.1.1. Evolució històrica

L'home ha tingut des de sempre la necessitat de tenir representacions del seu entorn que poguessin respondre a les seves preguntes, fent servir al llarg del temps les tecnologies i mètodes disponibles en cada moment històric. Si bé la cartografia ha estat la base de la representació geogràfica d'objectes, no es poden considerar els primers SIG, ja que només aportaven informació sobre situació,

El primer ús d'un SIG, encara que molt primerenc, el trobem a l'any 1854 per part del doctor *John Snow* **[1]**, pioner de la epidemiologia, que va cartografiar la incidència de casos de còlera al districte londinenc del *Soho*, per localitzar la font de la malaltia en un pou d'aigua contaminada. Va ser la primera vegada que un sistema d'informació geogràfica va se utilitzada per extreure coneixement que no fos purament geogràfic.

Els SIG, tal com els coneixem avui en dia, van tenir la primera utilització real a l'any 1962 a *Ottawa* (Canada) per part del departament federal de Silvicultura i desenvolupament rural, qui va desenvolupar un sistema, el CGIS (*Canadian Geographic Information System*) **[2]** que permetia gestionar els recursos naturals i usos del sòl del Canada d'una forma sistemàtica i georeferenciada.

A la mateixa dècada, *Howard T. Fisher*, va crear a la Universitat de *Harvard* el laboratori de Computació Gràfica i Anàlisi Espacial, a on van néixer els principals conceptes teòrics sobre els sistemes SIG, i van desenvolupar els primers programaris i sistemes que feien possible la seva explotació com ara SYMAP, GRID o ODYSSEY. Aquestos sistemes, han estan la base a partir de la qual han anat creixent els SIG, tant per usos comercials com per usos d'investigació.

El següent pas de la evolució dels SIG, ve de la ma d'empreses tals com Intergraph ESRI **[3]** o CARIS, qui a partir del sistema CGIS i les bases teòriques aportades per la Universitat de *Harvard*, desenvolupen el que seria la segona generació, clarament enfocada a un ús

comercial, introduint conceptes com la separació de la informació en capes, l'assignació d'atributs als elements geogràfics, i permetent la seva persistència en bases de dades.

La dècada dels 70 i principis dels 80, es caracteritza per l'entrada massiva d'entitats governamentals i administracions per desenvolupar els seus propis GIS que donessin resposta a les seves necessitats específiques.

Als 80 i 90, les empreses van tornar a agafar la iniciativa veient les oportunitats de negoci impulsades per les administracions públiques, i les possibilitats de desenvolupar eines informàtiques amb certa rapidesa, es van llençar a construir sistemes que poguessin ser aprofitades per la indústria en general. És el moment en que qualsevol empresa amb necessitats d'informació geogràfica tenia al seu abast un gran nombre de productes per escollir

A la dècada dels 90, amb la entrada a un preu assequible de maquinari al públic en general, el programari SIG, va entrar amb força en l'àmbit de l'usuari domèstic

Actualment, els SIG han evolucionat cap a un entorn distribuït, degut a la popularització d'Internet que permet crear sistemes descentralitzats i sobre tot a la aparició del fenomen del programari lliure fet que permet a una multitud de desenvolupadors col·laborar per crear sistemes vius a on tothom pot participar en el seu desenvolupament i ús.

2.1.2. Elements d'un SIG

Els SIG, estan formats per un seguit d'elements que interaccionen entre ells per a poder donar respostes a les funcionalitats requerides. Passem a descriure breument aquests elements d'un sistema SIG convencional:

- **Maquinari**: Entre el maquinari necessari per a construir un SIG, bàsicament calen servidors, que poden ser de bases de dades, de serveis web, d'imatges,.... Altres components de maquinari, podrien ser elements d'entrada i sortida de dades com ara impressores, digitalitzadors, equips GPS, equips de teledetecció...
- **Programari**: El programari SIG, està composat per un conjunt d'eines que permetran al sistema treballar d'una forma integrada per tal de poder emmagatzemar dades, manipular-les i oferir una interfície gràfica si és el cas.
- **Dades**: Possiblement és l'element crític de tots els SIG, ja que de la qualitat d'aquestes dependrà el correcte funcionament del conjunt. Les dades geogràfiques i les dades descriptives poden ser recollides o bé comprades a algun proveïdor. La majoria de SIG utilitzen un SGBD per crear i mantenir una base de dades amb la finalitat d'organitzar, manipular i documentar aquestes dades.
- **Personal**: La tecnologia SIG no té sentit sense el personal que gestiona el sistema per poder explotar-lo. Aquest personal el formen des del tècnic especialista que dissenya i manté el sistema fins a l'usuari final.

- **Procediments**: Un SIG opera segons el seu disseny i les seves regles de negoci. A partir del desenvolupament de procediments que actuen sobre el disseny i regles de negoci es pot obtenir un conjunt d'informació que pot ser molt valuós, per exemple, a l'hora de prendre decisions.

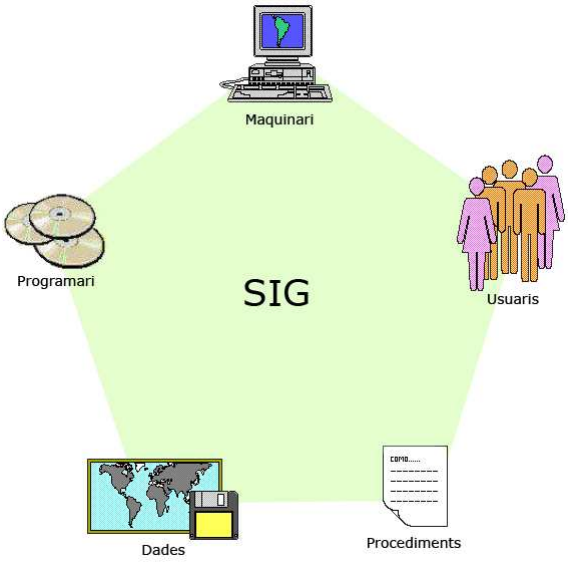


Figura 2.2 Elements d'un SIG

2.1.3. Tipus de dades manipulades pels SIG

Les dades que un SIG manipula són de dos tipus: dades descriptives i dades espacials. Les dades descriptives associen atributs de molt diferents tipus a un element geogràfic, dotant-lo de característiques qualitatives, com per exemple la composició geològica d'un àrea concreta.

Les dades espacials s'encarreguen de situar els elements geogràfics dintre d'un espai de representació concret, és a dir, situar-lo a la cartografia mitjançant un sistema de coordenades.

Dintre de les dades espacials, existeixen dos models diferents per representar-les:

Model vectorial

En el model vectorial els objectes del món real es representen mitjançant punts, línies i polígons. Aquesta representació no ha de ser obligatòriament única, ja que les dades que representen la realitat física, poden fixar-se en diversos models vectorials. Així, en una aplicació SIG per traçar rutes mínimes per carretera, una via podria ser representada simplement per una línea, mentre que per una aplicació SIG de topografia orientada a la gestió del territori podria ser representada per un àrea.

El model vectorial és útil quan s'han de descriure objectes geogràfics de forma precisa i compacte dels objectes representats, ja que encara que les dades siguin més difícils de

tractar, ofereixen possibilitats de manipulació molt altes.

Model raster

El model raster basa la seva funcionalitat en una concepció implícita de les relacions de veïnatge entre els objectes geogràfics. Consisteix en dividir la zona de representació en una retícula o malla de cel·les (píxels) i atribuir un valor numèric a cada cel·la com a representació del seu valor temàtic. Com que la malla és regular, la grandària del píxel és constant i es coneix la posició en coordenades del centre de les cel·les. Així doncs, es pot dir que tots els píxels estan referen ciats a uns sistema de coordenades concret.

Per tenir una descripció precisa dels objectes geogràfics continguts a la base de dades, la grandària del píxel s'ha de reduir en funció de l'escala, la qual cosa farà que la malla tingui una resolució alta. Tot i així com més gran sigui el número de files i columnes de la malla, major serà l'esforç en el procés de captura de la informació i major el cost computacional en el moment de processar-la.

El model de dades raster és útil quan s'han de descriure objectes geogràfics amb límits difusos, com ara l'estudi de la situació d'accidents geogràfics mòbils com les dunes, o la representació de les temperatures terrestres

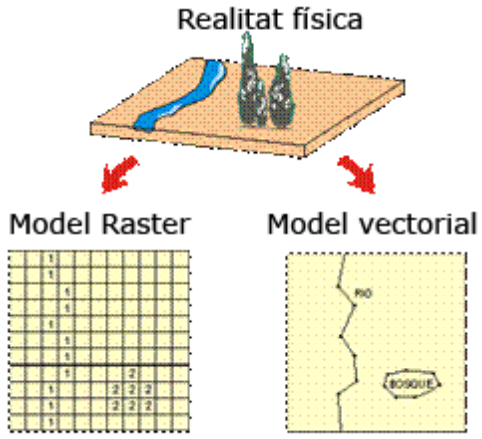


Figura 2.3 Models Raster i vectorial

2.2. SIG sobre web

El sorgiment de la revolució Web 2.0, ha fet que Internet deixi de ser exclusivament una eina de publicació de continguts, per passar a ser una plataforma a través del qual es poden desenvolupar aplicacions molt més complexes.

Aquesta revolució ha estat aprofitada d'una forma molt especial per certs sistemes SIG, per a poder arribar-hi a tothom, sense requerir unes condicions massa restrictives per part dels usuaris. Amb un ordinador personal mitjà, un programari mínim i una connexió a Internet, qualsevol pot accedir als sistemes GIS públics.

El paradigma d'aquesta revolució SIG sobre web ha estat *Google*, encara que no hagi estat l'única, ha sigut la que ha arribat amb més força, amb un model completament obert, tant al públic final com al de les comunitats desenvolupadores de programari.

2.2.1. *Google Earth* i *Google Maps*

Google, coneguda empresa d'Internet que va començar amb el seu cercador, està obrint contínuament el seu àmbit de negoci cap altres àrees, entre aquestes els SIG d'ús massiu.

Gràcies als seus acords amb TeleAtlas, Navteq i DigitalGlobe, ha aconseguit disposar d'una cartografia global de gran qualitat i d'un nivell d'actualització més que suficient per les aplicacions que volia desenvolupar. Aquest és un dels punts més destacables, ja que disposar de la cartografia mundial ha fet molt atractives les seves eines per als usuaris domèstics, doncs els permetia tenir un atles mundial en línea.

Google Earth, va néixer de la compra per part de *Google* de la empresa Keyhole, que ja disposava d'un programari similar, però amb una cartografia reduïda. Aquest programari, és local, és a dir s'ha d'instal·lar a cadascuna de les màquines a on es vol utilitzar, però la cartografia es en línea, ja que a mesura que ens movem pels mapes, es van demanant les parts que falten, de la mateixa forma que al fer apropaments, aconseguint una sensació de continuïtat i rapidesa molt apreciada pels usuaris.

Una altra característica molt apreciada pels usuaris, és la possibilitat que ofereix de veure la cartografia en tres dimensions, pel que es pot apreciar la orografia planetària. A més a més incorpora multitud de funcionalitats de geosituació, cerca de llocs específics, o incorporació de dades d'altres usuaris.

Figura 2.4 Google Earth

Google Maps, va ser el següent pas, prescindir del programari instal·lat localment, i de la mateixa forma que es descarrega en línea la cartografia, també es descarrega en línia el programari. L'únic requisit necessari per a l'usuari, és tenir un navegador web i connexió d'Internet. Entre les funcionalitats que es van incorporant a *Google Maps,* es podrien destacar la cerca d'adreces o darrerament Street View, una aplicació que permet veure certs recorreguts per ciutats en visió de 360 graus.

Figura 2.5 Google Maps

2.2.2. API i *Mashups*

L'èxit de *Google Maps*, ve en gran part per l'ús d'una cartografia global, un bon rendiment, però sobre tot de la filosofia d'obertura a funcionalitats de tercers.

Google, ha decidit publicar una API (Application Programming Interface) del seu *Google Maps* **[4]** , per tal que la comunitat de desenvolupadors afegeixin funcionalitats a sobre. Aquesta API, fa una descripció exhaustiva de totes les classes i mètodes públics que es poden fer servir per part dels desenvolupadors per a crear noves aplicacions utilitzant tots els recursos proporcionats per *Google Maps*.

Aquesta forma de treball, per la qual els desenvolupadors construeixen aplicacions aprofitant serveis d'altres, s'anomena Mashup, i la resposta a aquesta proposta de col·laboració, ha estat massiva. Actualment es conten per milers les utilitats que els usuaris han desenvolupat, com ara publicació de fotografies ge localitzades, rutes turístiques, o d'altres més complexes com jocs en línia basats en cartografia o directoris professionals.

Figura 2.6 Mashup Xing.com

2.3. Tecnologies involucrades

Moltes són les tecnologies necessàries per a poder realitzar uns sistema GIS sobre la web, passarem a continuació a descriure les més rellevants.

2.3.1. XML (*eXtended Markup Language*)

El llenguatge XML o llenguatge estès de marques, desenvolupat pel World Wide Web Consortium (W3C), és una simplificació del SGML (Standard Generalized Markup Language) de la mateixa forma que llenguatges de marques com HTML. Bàsicament un document XML serveix per portar dades estructurades de la mateixa forma que una base de dades, però amb unes regles auto incloses i sobre una codificació estàndard.

Les característiques més importants de XML són:

- Es un llenguatge extensible, és a dir, defineix una forma de treballar que en qualsevol moment pot ser ampliat amb semàntiques pròpies. Podem utilitzar XML per a intercanviar dades que hem definit per a un fi determinat.
- La forma d'analitzar un document XML és independent de les seves extensions, per tant el programari per a analitzar-lo es estàndard i independent dels tipus de dades que transporta.
- És llegible, ja que encara que no es dissenyat per a un processament humà, la seva

estructura és fàcilment comprensible, i està codificat en Unicode, per tant accepte la majoria de llengües.

Les parts d'un document XML són les següents:

- **Pròleg**:
  - Declara el document com a XML indicant la seva versió i codificació.
  - Referència al tipus de document que conté (DTD).
  - Comentaris i instruccions de processament.
- **Cos:** El cos d'un document XML, ha de tenir obligatòriament un únic element arrel, dintre del qual hi ha més elements en forma d'arbre.

Cadascun dels elements XML, poden tenir atributs, elements predefinits i dades. Aquests elements tenen unes regles molt ben definides per a ser ben formats.

Els requisits d'un document XML han de ser els següents:
Ben format: S'han de seguir les regles XML sobre els elements, la seva sintaxi i la seva correcció.
Vàlid: A part de ser correcte en la forma, ha de ser correcte n el contingut que transporta d''acord al DTD al que fa referència i que especifica els elements acceptats, els seus valors i la seva precedència a l'arbre.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE Edit_Mensaje SYSTEM "Lista_datos_mensaje.dtd"
                          [<!ELEMENT Edit_Mensaje (Mensaje)*>]>

<Edit_Mensaje>

     <Mensaje>

          <Remitente>
               <Nombre>Nombre del remitente</Nombre>
               <Mail> Correo del remitente </Mail>
          </Remitente>

          <Destinatario>
               <Nombre>Nombre del destinatario</Nombre>
               <Mail>Correo del destinatario</Mail>
          </Destinatario>


          <Texto>
               <Asunto>
                    Este es mi documento con una estructura muy senc:
                    no contiene atributos ni entidades....
               </Asunto>
               <Parrafo>
                    Este es mi documento con una estructura muy senc:
                    no contiene atributos ni entidades....
               </Parrafo>

          </Texto>

     </Mensaje>

</Edit_Mensaje>
```

Figura 2.7 Document XML

2.3.2. JavaScript

Javascript és un llenguatge de programació interpretat orientat a objectes i basat en Java.

Aquest llenguatge, és el més utilitzat en els navegadors i la majoria disposen del seu intèrpret,

per tant és la millor opció a l'hora de dotar a les pàgines web d'una certa interactivitat.

Estem habituats a veure pàgines amb certs efectes gràfiques, o formularis que responen a certes restriccions d'introducció de dades, doncs bé, la majoria d'aquesta funcionalitat està construïda a sobre de JavaScript.

Encara que pugui semblar que les funcionalitats que es poden obtenir d'un llenguatge interpretat no siguin gaire potents, si es barregen amb l'accessibilitat als elements del document HTML amb el qual es descarreguen, fent servir DHTML (HTML dinàmic), es pot aconseguir un molt complet comportament de la interfície gràfica d'usuari, i potents prestacions de càlcul.

De fet *Google Maps* fa servir unes llibreries Javascript no incloses a les pàgines, si no que es descarreguen a partir d'aquestes, que ofereixen tota la funcionalitat de la aplicació. Cada vegada es veuen més llibreries especialitzades en creació de components d'interfície d'usuari, que proporcionen components molt complexos per desenvolupar aplicacions complertes basades en Javascript.

2.3.3. AJAX (Asynchronous JavaScript And XML)

Si agafem aquests components Javascript que poden construir una aplicació complerta, i fem servir XML com a mitjà de comunicació entre aplicacions o entre aplicacions i servidors, obtenim una potent aplicació distribuïda, això és bàsicament AJAX, mesclar aquestes dos

L'usuari d'una aplicació basada en AJAX, pot gaudir d'una aplicació web tenint la sensació d'estar treballant amb una aplicació d'escriptori, ja que una vegada carregada la pàgina HTML i les seves llibreries Javascript, la gran majoria d'interaccions de dades que faci amb altres usuaris o amb el servidor es faran intercanviant missatges XML en segon pla sense tenir que recarregar la plana com es feia en aplicacions no AJAX.

Aquesta tècnica permet apropar les aplicacions basades en web al funcionament i aparença de les aplicacions d'escriptori, formant part de les aplicacions anomenades RIA, o Aplicacions Riques d'Internet.

2.4. OGC i estàndards

L'OGC (*Open Geospatial Consortium*) va ser creat al 1994 agrupant a més de 250 organitzacions públiques i privades amb l'objectiu de definir un seguit d'estàndards oberts i interoperables en l'àmbit dels SIG.

Històricament cada implementació de SIG que s'han dissenyat, oferien funcionalitats específiques i protocols propietaris, de forma que era pràcticament impossible compartir dades o fer que sistemes diferents col·laboressin entre ells. Aquest fet feia que els usuaris dels sistemes estiguessin captius per les inversions fetes i no els permetia avançar amb prou seguretat.

Els estàndards oberts i interoperables, permeten que els programaris es puguin entre si, al compartir dades en uns mateixos formats i amb uns mètodes de comunicació comuns. Avui en dia la immensa majoria de programari segueix aquests estàndards contribuint a crear sinèrgies en el món del SIG.

Entre les contribucions més importants que l'OGC ha impulsat podem destacar GML, WMS i WFS.

2.4.1. GML (*Geography Markup Language*)

GML és un estàndard de modelatge, transport i emmagatzemament de dades geogràfiques que mitjançant un document XML permet que els sistemes SIG puguin intercanviar dades o col·laborar entre ells.

Aquest llenguatge, està basat en Esquemes XML, una evolució dels DTD però construït també en XML, a on s'especifiquen com seran les dades, quines regles hauran de seguir els seus element...

GML defineix dos conceptes principals:

- **Entitats**: Que representen objectes de la realitat física com ara vies de comunicació o edificis
- **Objectes geomètrics**: que defineixen una localització o regió.

Les entitats es poden relacionar amb objectes geomètrics que les localitzin, fent que sigui un sistema molt versàtil, ja que podem ubicar objectes físics dintre de regions diferents, com ara un poble dintre d'una comarca o dintre d'una província.

GML, també té la capacitat d'integrar dades de diferent naturalesa com objectes discrets, dades vectorials, o fins i tot dades raster.

2.4.2. WMS (*Web Map Service*)

WMS, és un servei definit per l'OGC que permet produir imatges de dades espacials de forma dinàmica a partir d'una informació geogràfica. A través de la invocació d'una URL (*Uniform Resource Locator*), amb els paràmetres adequats que solen incloure la informació a visualitzar el sistema de coordenades a emprar i d'altres especificacions, el servidor retorna una imatge en formats gràfics específics per ser visualitzats en pantalles d'ordinador, que representa la informació sol·licitada.

Aquesta forma d'interaccionar entre clients i servidors, permet construir xarxes de servidors WMS, que contribueixin al desenvolupament del sistemes GIS distribuïts a on cada servidor, per exemple es faci responsable d'un àrea de representació.

2.4.3. WFS-T (*Web Map Service Transaction*)

Aquest estàndard OGC, permet a través de GML, editar dades geogràfiques.

Google Maps té una funcionalitat similar, ja que es capaç de deixar que l'usuari editi punts línies i àrees, però no és possible d'emmagatzemar-les, ja que no ofereix el servei de persistència. L'estàndard WFS-T, defineix un document amb el qual un client pot enviar a un servidor que suporti l'estàndard,per que sigui aquest el que ofereixi aquesta capa de persistència.

Les possibilitats que ofereix aquest estàndard són les següents:

- Crear noves entitats geogràfiques.
- Obtenir una descripció de les propietats de les entitats.
- Esborrar entitats.
- Modificar entitats ja existents.
- Bloquejar entitats per a que un usuari les pugui manipular. Aquesta característica li dóna la característica de transaccional.

La comunicació entre servidor i client, sempre a petició del client, es fa amb una petició HTTP, i el servidor respon amb un document GML que el client haurà de llegir, extreure les informacions necessàries i representar-les o manipular-les.

La comunicació en sentit client cap a servidor WFS-T, es pot fer de dos formes:

- A través d'intercanvi de documents XML amb el mètode POST d'HTTP, o a través de SOAP.
- A través del mètode GET d'HTTP codificant la informació com a paràmetres de la URL.

3. DESENVOLUPAMENT DEL TREBALL I PRODUCTES OBTINGUTS

En aquest capítol, farem el desenvolupament pròpiament dit de les llibreries *Javascript* necessàries per a poder dotar de la funcionalitat requerida, i així oferir a la comunitat una plataforma per a construir *mashups* amb possibilitats d'edició i persistència d'entitats geogràfiques.

Començarem amb una descripció del sistema proposat, el seu funcionament, i les relacions entre els diferents components que el conformen.

Continuarem amb la explicació detallada de les classes implementades, tant en la part de model com en la de controlador. Es farà una descripció dels processos de comunicacions entre les llibreries implementades i el servidor WFS-T, amb la que es podrà veure quines interaccions i en quin ordre es produeixen.

Per últim, farem únicament una descripció de la part de la vista, ja que degut a les limitacions d'espai i a la complexitat de les llibreries d'IGU escollides, seria molt extens fer una explicació detallada de la programació feta, deixant la demostració de les utilitats implementades per a la presentació en vídeo del treball.

3.1. Descripció general del sistema implementat

Existeixen moltes possibilitats en el disseny global del sistema, que dependrà de la disponibilitat dels elements necessaris, dels coneixements previs, i sobre tot del temps requerit per a instal·lar-los i configurar-los correctament.

El sistema proposat per a fer la implementació, consta dels següents elements:

- ***Tomcat***: Serà el nostre servidor *http*, que oferirà les pàgines web, llibreries Javascript dissenyades, i d'altres recursos necessaris que per a la construcció de *mashup*s per part de tercers.
- ***Geoserver***: Servidor *WFS-T*, que aportarà les funcionalitats d'interoperabilitat entre els clients i el servidor de persistència. La seva funció principal és la de traduir les peticions *GML* entre els clients i la capa de persistència, oferint una forma clara i neta de relació.
- ***PostgreSQL+PostGIS***: Sistema de gestió de base de dades que haurà d'oferir la funcionalitat de persistència adaptada als requeriments de l'estàndard *GML*, per aquest motiu, necessitarà d'un mòdul d'adaptació per fer que el model de dades sigui compatible amb els estàndards *OGC* emprats.

També podem considerar com a part del sistema general altres elements que si no hauran estat dissenyats per nosaltres, ens oferiran els serveis necessaris per a completar-lo

- ***Maps.google.com***: Servidor de *Google* que ens proporcionarà tant la cartografia necessària com les llibreries sobre les quals construirem la nostra.
- **Usuari**: Per tal de poder il·lustrar el funcionament del sistema, cal que el tinguem present, doncs bàsicament és l'usuari qui interacciona amb els altres elements del sistema.

Tots aquest elements estaran comunicats a través de la xarxa d'Internet, encara que no tots tindran un caràcter públic, ja que per exemple la base de dades, no serà accessible directament pels clients, si no que serà el servidor *WFS-T* qui s'encarregarà d'accedir-la.

En l'esquema proposat, el servidor http i el servidor *WFS-T*, s'han disposat en màquines diferents, però això només és una possibilitat, doncs en el nostre cas particular residiran a la mateixa màquina.

Una altra possibilitat, en el cas de que es facin servir les llibreries dissenyades per terces, hauria proveir d'un segon servidor *http* a on residirien les noves funcionalitats o utilitats construïdes a sobre de la nostra, és a dir, s'actuarien com a repositoris de *mashups*.

El sistema proposat restaria d'aquesta forma:

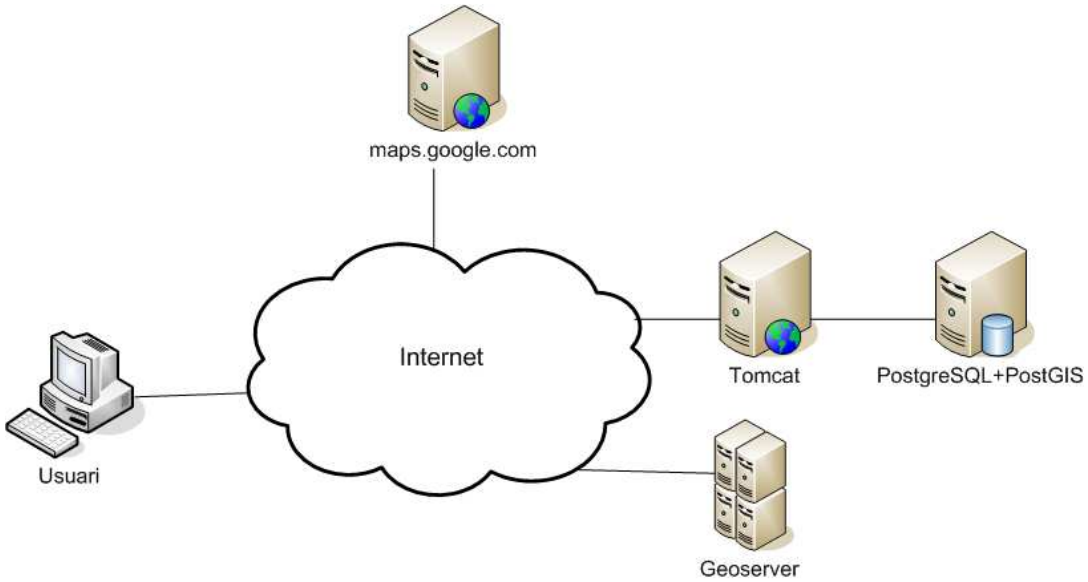


Figura 3.1 Components del sistema

3.1.1. Càrrega de la pàgina web

El primer pas per a poder fer servir el sistema per part d'un usuari, seria la càrrega de la pàgina principal de la aplicació.

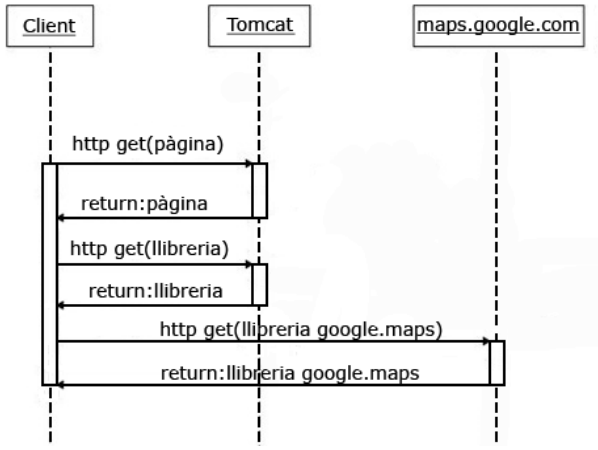


Figura 3.2 Diagrama de seqüència de la càrrega de pàgina

La seqüència seria la següent:

1. El client demana a través d'una petició *http get* la plana principal al servidor *Tomcat*, que és a on resideixen tots els recursos dissenyats.
2. El servidor *http*, respon amb la pàgina demanada.
3. El client analitza la pàgina rebuda, i comprova la existència de referències a llibreries *Javascript* que encara no s'ha descarregat.
4. El client demana al servidor *Tomcat* la llibreria o llibreries dissenyades.
5. El servidor *http* respon amb un document a on està la llibreria demanda.
6. El client demana al servidor de mapes de *Google* la llibreria bàsica per a poder manipular les funcionalitats de l'API de *Google*, en aquest cas, al ser una llibreria llicenciada, s'ha de demanar amb una clau prèviament demanada.
7. El servidor de *Google* ens serveix les llibreries demanades.

En aquest punt ja disposem en el client de totes les parts necessàries per a poder fer servir les funcionalitats que s'han afegit a través de les nostres llibreries.

A partir d'ara la comunicació entre el client i el servidor, es farà sense demanar cap més recurs que no sigui en segon pla a través de peticions i respostes de documents *GML* al servidor *Geoserver*, o peticions d'imatges al servidor de *Google*.

A la descripció de les interaccions del sistema, obviarem totes les peticions a aquest servidor per tal de donar claredat al sistema.

3.1.2. Recuperació de dades

Una vegada descarregada la pàgina *web*, el *Javascript* inclòs a la pàgina haurà d'inicialitzar la nostra llibreria, i després demanar les dades a manipular al servidor *Geoserver*.

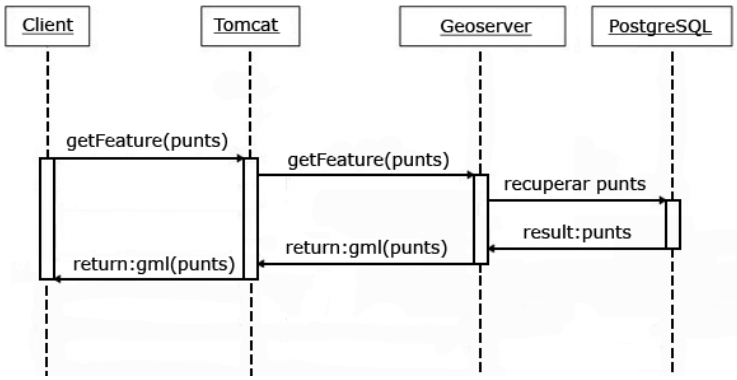


Figura 3.3 Diagrama de seqüència de la recuperació de dades

El procés seria el següent:

1. El client demana a través d'una petició *http* una entitat al servidor *Tomcat*, en aquest cas la entitat punts. Per a les línies i les àrees, seria similar.
2. El servidor *http*, que té instal·lat el servidor *Geoserver* com a una aplicació *J2EE*, s'encarrega de passar-li la petició.
3. El servidor *Geoserver*, analitza la petició, i decideix quina ha estat la entitat demanada. A través de la seva configuració, selecciona el magatzem de dades i la taula adequada per atendre la petició.
4. Recupera del servidor de base de dades la col·lecció de punts demanada, i construeix el document *GML* adequat, passant-li al servidor *Tomcat.*
5. El servidor *http* respon al client amb el document *GML* rebut.
6. Al rebre el client el document *GML*, s'encarrega d'extreure la col·lecció de punts demanada, i a través de la llibreria que ha fet la petició, dibuixa els punts en una capa que es mostrarà a sobre dels mapes proporcionats per *Google Maps*.

A partir del moment en el qual el client rep les dades del servidor, entra en escena l'usuari, qui és l'encarregat de afegir, modificar o esborrar les entitats recuperades. Tot aquest procés es realitza en l'àmbit local, és a dir, en el client, sense comunicació amb els servidors.

Només quan l'usuari doni per bones les modificacions fetes, llançarà una petició de modificació als servidors per tal de que les seves modificacions siguin persistents.

3.1.3. Modificació de dades

És en aquest punt a on entren en joc les capacitats específiques que realment estem implementant, doncs el més interessant és donar persistència a les dades.

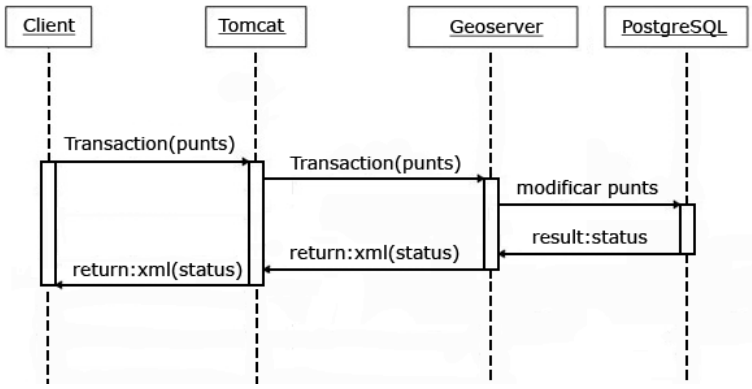


Figura 3.4 Diagrama de seqüència de la modificació de dades

Una vegada l'usuari dóna per bones les modificacions fetes en local, ha de fer que aquestes siguin persistents, per a poder-les recuperar en una altra sessió o que altres usuaris les puguin recuperar, per això es desencadena el següent procés:

1. El client envia al servidor *Tomcat*, un document *XML* a on s'especifica amb l'estàndard *WFS-T*, totes les modificacions sobre les entitats que desitgem fer. Per a poder fer això la nostra llibreria *Javascript*, ha de tenir constància de totes les entitats creades, modificades i esborrades respecte als recuperats anteriorment.
2. El servidor *http*, recull el document i li passa al servidor *Geoserver*.
3. El servidor *Geoserver*, analitza la petició, i comença a modificar les entitats a la base de dades amb les instruccions rebudes al document.
4. La base de dades, respon a cada una de les modificacions amb els resultats obtinguts, és a dir, si s'han pogut fer o no.
5. El servidor *Geoserver* composa un document *XML* amb el resultat de totes les operacions demanades, i si és el cas amb l'identificador de les entitats creades.
6. Aquest document es passat al servidor *http*, qui s'encarrega d'enviar-ho al client
7. El client haurà d'analitzar el resultat de les operacions demanades, i en conseqüència oferir a l'usuari missatges de conformitat, o prendre accions per resoldre els possibles conflictes trobats.

3.2. Llibreries Javascript implementades

En aquest apartat farem una descripció de les classes implementades per a poder dotar de la funcionalitat demanada al conjunt de llibreries. S'ha optat per fer una aproximació basada en el paradigma de la programació orientada a objectes, en endavant POO.
Encara que és una forma complexa d'afrontar un problema com el present, amb la necessitat d'implementació forçosa amb *Javascript*, es creu que és l'única que pot oferir garanties de prova, desenvolupament i manteniment suficients.

Una altra decisió presa, ha estat la incorporació del patró d'arquitectura de programari Model Vista Controlador **[13]**, en endavant MVC, ja que ens proporcionarà una separació clara entre el model de les dades, el seu funcionament, i la vista amb la qual la farem servir.

Aquesta separació entre el model i la interfície gràfica d'usuari, en endavant IGU, fa que els possibles usuaris de les nostres llibreries, puguin fer servir només el model, i construir la seva IGU, o simplement fer servir la totalitat de les llibreries, és a dir model i IGU conjuntament.

Per a poder unir les dos capes, s'ha implementat un seguit de classes adaptadores que permeten fer-les servir en el cas que ens ocupa d'interacció amb *Google Maps*, o desenvolupat unes diferents amb qualsevol llibreria gràfica que proporcioni l'accés als mapes, com podrien ser *Yahoo Maps*.

3.3. Model

S'ha seguit el model d'entitats geogràfiques simples que l'*OGC* proposa, encara que s'ha modificat el nombre de classes i dependències per tal de donar més claredat al disseny, doncs en cap cas farem servir tots els tipus d'entitats apuntats, i només ens referiren a punts, línies simples i àrees.

Podem veure a la següent figura totes les entitats bàsiques i les seves relacions en un diagrama *UML* de classes estàtica, i com les hem modificat per al nostre disseny, sobre tot eliminant classes que no representarem, o herències que no seguirem, ja que no ens faran falta una col·lecció tant gran d'entitats geogràfiques a representar.

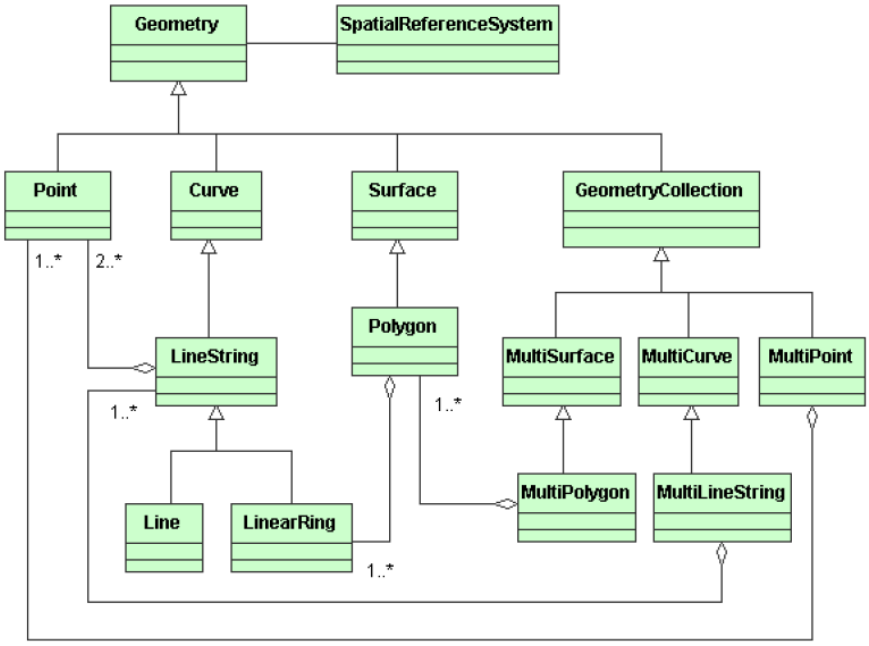


Figura 3.5 Diagrama UML estàtic de classes proposat per l'OGC

En el nostre cas, aquest diagrama restaria de la següent forma:

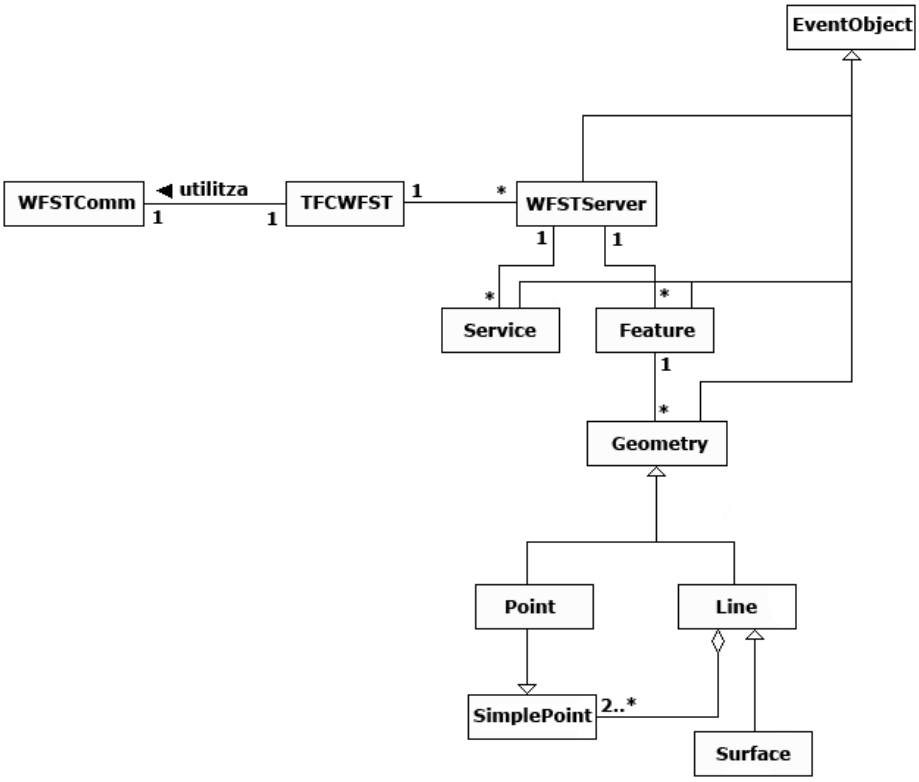


Figura 3.6 Diagrama parcial UML estàtic de classes dissenyat

Les classes que apareixen a la figura 3.6, representen una part del disseny, ja que falten les classes d'adaptació i les classes que composen l'IGU.

S'han agrupat segons les seves funcions

- Classes d'utilitat:
  - ***EventObject***: Classe que proporciona la capacitat de llançament i subscripció a events, implementant el patró observador-observable.
  - ***HashTable***: Classe d'utilitat que permet utilitzar taules amb clau d'accés.
- Classes geomètriques:
  - ***Geometry***: Classe abstracta, base pels elements geogràfics bàsics.
  - ***SimplePoint***: Classe que representa un punt.
  - ***Point***: Classe que representa un punt amb identificador d'entitat.
  - ***Line***: Classe que representa una línia.
  - ***Surface***: Classe que representa una superfície.
- Classes de representació d'objectes de servidor:
  - ***WFSTServer***: Classe que representa els servidors amb els que treballem.
  - ***Service***: Classe que representa un servei concret suportat per un servidor, amb el tipus de servei i les URL que el proporcionen.
  - ***Feature***: Classe que representa cadascuna de les capes que un servidor pot proporcionar, aquestes poden ser de tres tipus segons els tipus de dades geomètriques que continguin.
- Classes principals
  - ***TFCWFST***: Classe principal a través de la que accedim a tota la resta.
  - ***WFSTComm***: Classe d'utilitats de comunicacions, bàsicament la connexió amb el servidor a través d'*HTTP* amb els mètodes *GET* i *POST* i mètodes d'utilitat.

Passem a descriure-les breument agrupades en aquest mateix ordre, per tal d'obtenir una visió ascendent de la resolució del problema.

3.3.1. Classes d'utilitat: *EventObject*, *HashTable*.

Aquestes dos classes, encara que no formen part del model, són imprescindibles per a poder dotar de la funcionalitat necessària a les classes dissenyades. Només es farà una descripció de la seva funcionalitat, ja que són classes de domini públic.

Classe *EventObject*

Aquesta classe constituirà la base per a totes les classes que necessitin implementar el patró observable, que farem servir per a poder llençar events amb la informació necessària

El patró observable defineix una dependència un a molts de forma que quan un dels objectes observables canviï el seu estat els observadors d'aquest objecte seran notificats del canvi.
Bàsicament un objecte observable llença, cada vegada que es modifica el seu estat, un event amb el mètode *raiseEvent*, amb un nom d'event i una sèrie de paràmetres. Els objectes que estan interessats en rebre notificació, prèviament s'han subscrit a aquest event amb el mètode *attachEvent*, indican la funció que manegarà l'avís.

La implementació **[10]** ha estat recollida d'*Internet*, encara que s'ha hagut d'adaptar amb un atribut d'àmbit necessari per poder-lo fer servir a les classes de la vista.

Classe *HashTable*

Per a certes classes necessitarem contenidors d'objectes als que sigui fàcil interrogar per tal d'obtenir els objectes desitjats. Per a aquesta funció s'ha decidit fer servir la classe *HashTable* que ens permet associar una clau d'accés de tipus *String* a un objecte, i així poder incloure objectes referenciats per un identificador o un descriptor.

En aquesta ocasió la implementació **[10]** en *Javascript* s'ha obtingut d'Internet, sense necessitat de modificar-la.

3.3.2. Classes geomètriques: *Geometry*, *SimplePoint*, *Point*, *Line* i *Surface*

L'objectiu fonamental d'aquestes classes, és representar els objectes geomètrics que es recuperaran del servidor *WFS-T*, seran gestionats per les llibreries, i finalment es salvaran al servidor.
Passem a continuació a descriure cadascuna d'aquestes classes proporcionant una breu descripció, els seus atributs i els seus mètodes.

Classe *Geometry*

Aquesta classe és la base a partir de la qual s'estendran les classes de representació d'entitats geogràfiques pròpiament dites. Aquesta classe proporciona una implementació comuna i uns mètodes abstractes que hauran d'implementar els objectes heretats.
Cal notar que en *Javascript* no hi ha possibilitat de definir classes abstractes, per tant només es descriurà el mètode sense tenir implementació ni definició a la classe *Geometry*.
L'objectiu de definir mètodes abstracte és proporcionar una interfície comuna per a que les altres classes les puguin manipular d'una forma homogènia sense saber de quin tipus d'objecte es tracta en concret.

Classe *SimplePoint*

La classe *SimplePoint* representa un punt geogràfic definit per la seva latitud i la seva longitud exclusivament. S'ha diferenciat entre aquesta classe i la classe *Point*, per motius de claredat, doncs les classes *Line* i *Surface*, faran servir punts per definir els seus vèrtex, i aquest punts no han de proporcionar identificadors d'entitat únics. Bàsicament la diferència entre un *SimplePoint* i un *Point* està en l'atribut *id*.

Classe *Point*

La classe *Point* representa la entitat geogràfica bàsica, i hereta de la classe *SimplePoint* i *Geometry*. No descriurem els mètodes i atributs de les seves classes pare, ja que ja han estat descrits.

Classe *Line*

La classe línia, és bàsicament un conjunt de punts ordenats que representen una successió de segments. Per tal de poder emmagatzemar aquest conjunt de punts s'ha fet servir una matriu de punts, ja que conserva l'ordre d'entrada dels objectes.

Classe *Surface*

Per últim la classe que representa un àrea, és simplement una extensió de la classe *Line*, ja que el seu comportament és exactament igual, amb la única diferència del mètode *toString*.

3.3.3. Classes de Representació de Servidor: *Server*, *Service* i *Feature*

Aquest grup de classes, tenen com a missió principal representar un servidor amb el que estem treballant, les seves capacitats, els serveis que presten i les seves col·leccions de capes.

Procés de càrrega de dades

Abans de tot s'explicarà el mecanisme pel qual a partir d'una *URL* de servei del servidor *WFST* i un objecte de comunicacions, es desencadena la càrrega de totes les dades i configuracions.

A la primera fase, un objecte *Server* llença una petició *GetCapabilities* a la *URL* de servei proporcionada, *Geoserver*, li retorna un document *XML* amb totes les seves capacitats i tipus de Features de les quals disposa.

En aquest moment, l'objecte *Server*, construirà tants objectes *Service* com missatges informi *Geoserver* que és capaç d'atendre, que bàsicament seran *DescribeFeatureType*, *GetFeature* i *Transaction*, amb les seves corresponents *URL*'s de servei.
De la mateixa forma, l'objecte *Server*, crearà tants objectes *Feature* com s'hagin declarat al document retornat per *Geoserver*, de tal forma que indicaran la disponibilitat d'aquestes.

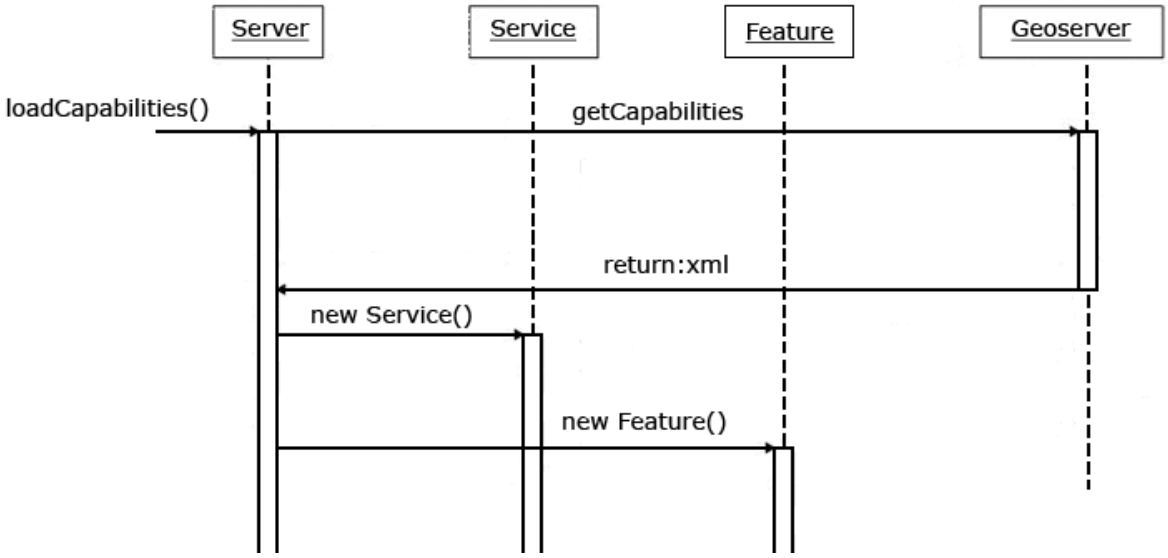


Figura 3.7 Diagrama de seqüència de la petició getCapabilities

En una segona fase, l'objecte *Server*, s'encarregarà de demanar informació més detallada de cadascuna de les *Features* creades, fent servir per això un dels objectes *Service* ja creats, en concret el que representi el servei *DescribeFeatureType*.
La resposta a cadascuna de les peticions, farà que s'aconsegueixi més informació sobre cada *Feature*, la més important de totes, el tipus de dada geogràfica de la que disposa, per tal de determinar si és susceptible de ser tractada per les nostres llibreries (punts, línies i àrees)

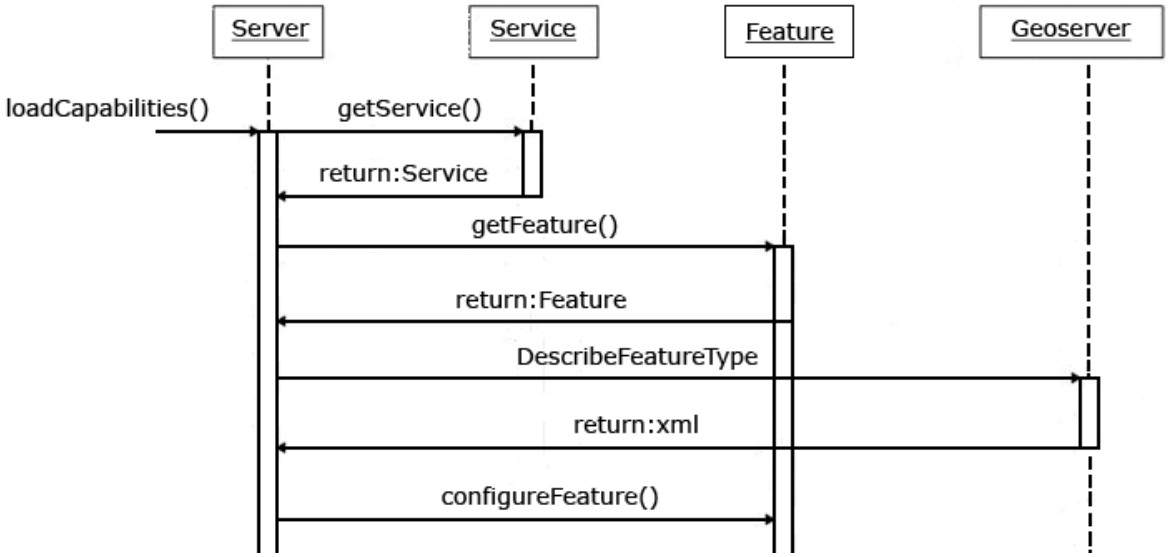


Figura 3.8 Diagrama de seqüència de la petició DescribeFeatureType

Per últim, una vegada ja tenim totes les capacitats del servidor i configurades cadascuna de les *Features* disponibles, es decideix quines d'aquestes caldrà recuperar.
Com no totes les *Features* disponibles poden ser manipulades per les nostres llibreries, només seran vàlides les que contemplin els tipus de dades geogràfiques adequades, deixant que l'usuari seleccioni d'aquestes les que vol recuperar completament.

Per a cadascuna de les *Features* seleccionades, es genera un missatge *GetFeature* que el servidor recuperarà de la seva base da dades i la retornarà a l'objecte *Server*. De la mateixa forma que en la situació anterior, el *Server* recuperarà el *Service* necessari per a la petició, fent servir les seves *URL*'s per a la comunicació.

L'objecte *Server*, serà l'encarregat de crear la col·lecció d'objectes *Geometric* adequada i donar-li a la *Feature* a la que pertany per a una posterior gestió.

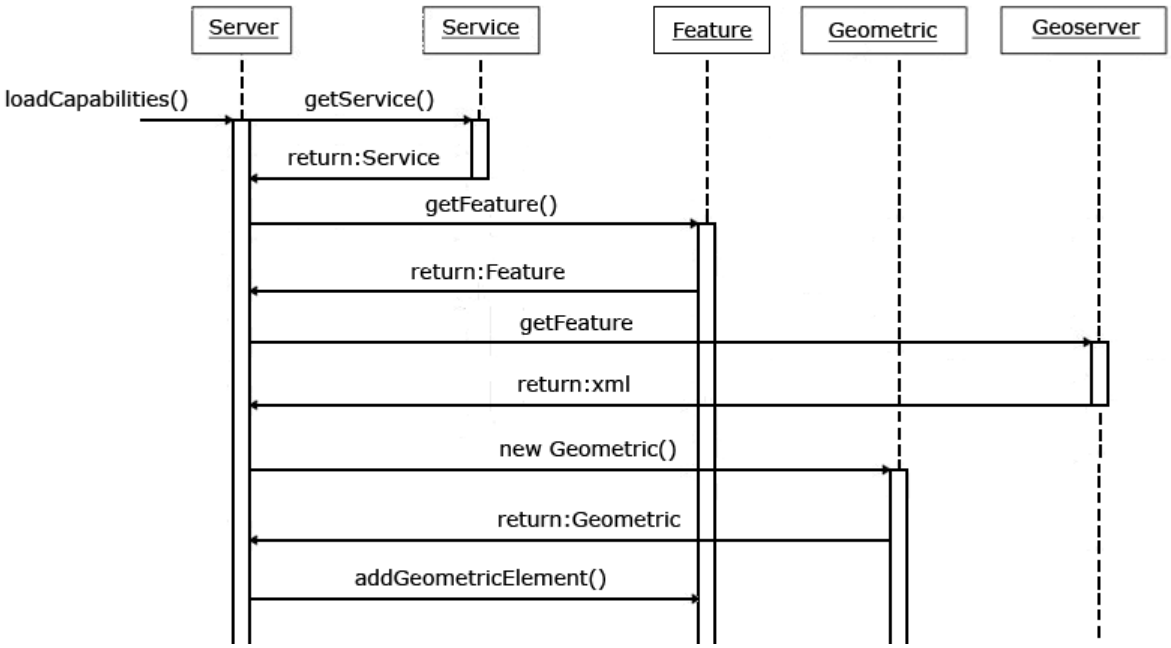


Figura 3.9 Diagrama de seqüència de la petició getFeature

Classe *Server*

Amb aquesta classe es representa el servidor i les seves característiques, oferint mètodes per recuperar aquestes, així com les dades de cada una de les features seleccionades.

Al definir una classe Server, podem tenir més d'un objecte d'aquesta classe, el que ens permetrà poder recuperar capes de diferents servidors, i poder-les gestionar conjuntament, com si fossin recuperades d'un mateix servidor. No cal dir que a l'hora de salvar les dades de cadascuna de les capes, es salvaran al servidor des de s'hagin descarregat.

Aquesta possibilitat de treballar amb més d'un servidor, ens podria servir per a copiar dades

geomètriques d'un servidor en un altre, o recuperar dades des de servidors amb diferents finalitats i poder-les veure totes juntes en una mateixa interfície d'usuari.

Classe *Service*

Cada servidor, ofereix un seguit de primitives *WFS-T* que s'hauran de conservar per tal de poder fer la gestió de les dades d'una forma correcta, bàsicament cadascun dels serveis oferts (modificació, esborrat...), poden tenir adreces *URL* diferents de la del servidor *WFS-T*, i a més a més depenent del mètode *HTTP* emprat, també poden ser diferents.

Cal tenir constància de quines adreces tenen cadascun dels serveis per tal de dirigir-se correctament en les peticions que es facin, ja que, encara que les adreces solen ser les mateixes, es pot trobar configuracions de servidors en que no ho siguin.

Classe *Feature*

La classe Feature, és la més important d'aquest grup, ja que la seva missió principal, és la d'agrupar els elements geogràfics recuperats de cada capa, oferint tots els mètodes necessaris per a la seva gestió.

Aquestes capes, fonamentalment gestionaran les dades geomètriques de la capa, permetent modificar, afegir i esborrar elements, i tenir constància dels canvis realitzats per a poder fixar les modificacions en els servidors.

És important destacar la gestió de elements esborrats, doncs, s'haurà de tenir sempre constància de quins han estat durant la execució de l'aplicació per donar l'opció a l'usuari de recuperar-los abans de salvar-los definitivament.

3.3.4. Classes principals; TFCWFST i WFSTComm

Aquestes dos últimes classes, estan separades per a diferenciar les seves funcions, ja que una simplement serà la classe contenidora de la resta, i l'altre contindrà tota la funcionalitat de comunicacions, ocupant-se de enviar i rebre els documents XML que li demanin.

Classe *TFCWFST*

Aquesta classe, conté tots els elements necessaris per a poder fer funcionar la llibreria, la llista de servidors i un objecte de comunicacions per a poder interactuar amb els servidors *WFS-T*.
Simplement es tracta de la classe que haurem de crear per a poder fer servir tota la resta, i ens permetrà construir més d'un objecte per disposar de més d'una instància del programari complert.

Classe *WFSTComm*

Tota la funcionalitat de comunicacions s'ha unificat en aquesta classe, de forma que és la que

proporciona les diferents connexions i és capaç de construir els objectes necessaris segons els documents *XML* retornats.
Si es volgués modificar el comportament de la connexió, o variessin els protocols, com per exemple amb un nou estàndard *OGC*, només caldria modificar aquesta classe, i la resta de classes podrien seguir funcionant correctament.

3.4. Vista

La vista ofereix un seguit de recursos gràfics per fer que la relació de l'usuari amb el programari sigui la més fluida possible. Per tal d'aconseguir aquest objectiu s'han fet servir unes llibreries *Javascript* que proporcionen un conjunt de components molt similars als que es fan servir per dissenyar aplicacions de sobretaula amb les que els usuaris solen estan molt familiaritzats. En concret s'han fet servir les llibreries *extjs* **[16]**, que ens donen aquestes possibilitats.

Podem dividir la vista en dues parts:

- Interfície d'interacció directa amb les dades, constituïda per uns components de selecció i visualització estàndards:
  - Finestra principal.
  - Menú.
  - Finestra de càrrega.
  - Explorador de capes i objectes geomètrics.
- Interfície d'interacció geogràfica, que permeten la gestió del elements geomètrics directament damunt dels mapes proporcionats per *Google Maps*.

Encara que totes dues parts formin part de la mateixa interfície gràfica, les seves funcions són molt diferents, ja que la interfície d'interacció amb les dades, treballa directament amb el model i la interfície d'interacció geogràfica treballa amb la capa de controlador per accedir al model.

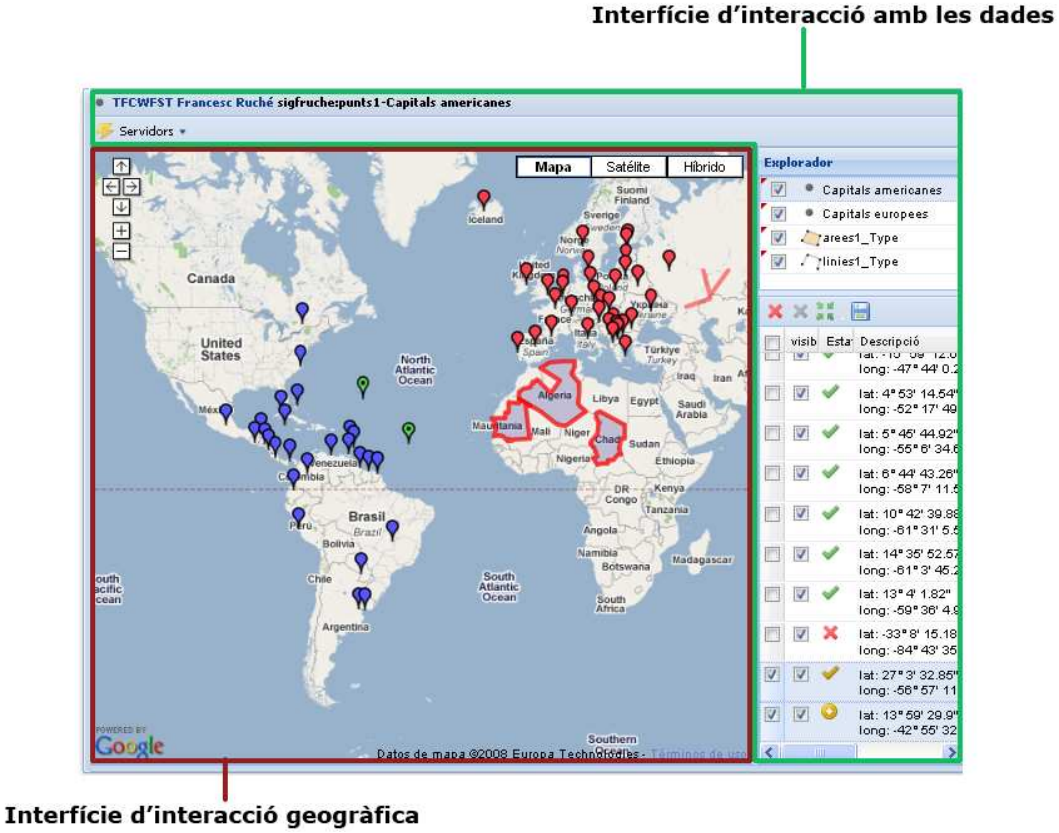


Figura 3.10 Divisions de la interfície gràfica

3.4.1. Interfície d'interacció directa amb les dades

Aquest part de la interfície gràfica, permet accedir a les dades geomètriques a través de components gràfics comuns, com ara menús, llistes, seleccions, botons d'acció...
Passarem a descriure breument les funcionalitats d'aquesta interfície:

Finestra principal
Tota la interfície gràfica està continguda dintre d'una finestra que es pot moure per tot el navegador web i també modificar la seva grandària per adequar-la millor a l'espai de visibilitat.

Figura 3.11 Barra de títol

Cal destacar que a la barra de títol, s'ofereix informació del programari utilitzat, així com de la capa actualment seleccionada, indicant amb una icona quin tipus de dada geogràfica conté.
També disposa del botó estàndard de tancament de la finestra, que iniciaria una acció de tancament assegurant que les capes modificades, si existeixen, poden ser salvades abans del tancament a voluntat de l'usuari.

Menú

Figura 3.12 Menú

Dintre de l'únic menú del que disposem, podem fer les següents accions:

- **Nou Servidor**: Accedir a un servidor al que no hem accedit encara, i poder inspeccionar els serveis, característiques i capes de dades que ens ofereix. Si intentéssim accedir a un servidor al que ja hem accedit abans, se'ns indicaria que el recuperéssim des de l'opció Servidors.
- **Servidors**: Accedirem als servidors que ja hem inspeccionat per a poder carregar noves capes, o descarregar capes amb les que ja hem treballat.
- **Salvar dades**: Salvar totes les capes de dades que hagin estat modificades, independentment del servidor que las contingui.
- **Nova instància**: Crear una nova instància del programari, amb les mateixes funcionalitats, però sense l'estat actual, és a dir amb un estat inicial.
- **Sortir**: Abandonar la execució del programari, preguntant primer, en el cas d'haver

dades modificades, si volem salvar-les,

Finestra de càrrega

Si des del menú accedim tant a Nou Servidor, com a un servidor la inspeccionat des de l'opció Servidor, accediríem a una finestra a on tenim les dades del servidor i les capes que conté.

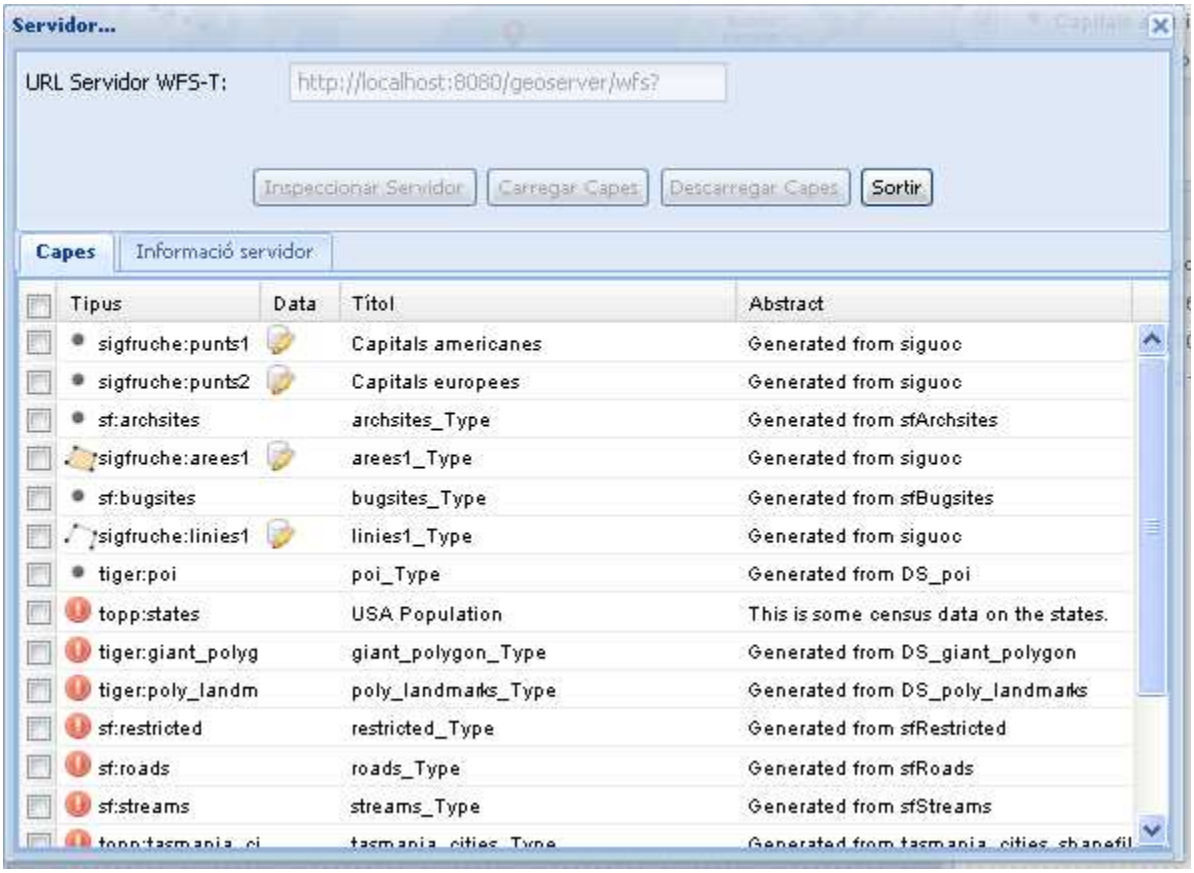


Figura 3.13 Finestra de càrrega

Aquesta finestra ens proporciona una caixa de text per a poder introduir l'adreça de servei del servidor, i posteriorment connectar-nos. Com a resultat de la connexió, podem accedir al conjunt de capes disponibles seleccionant les capes que volem recuperar.

Cadascuna de les capes, ve definida pel seu nom, el seu títol i la descripció detallada de la mateixa, així com una icona que representarà el tipus de dada geomètrica detectada (punts, línies o àrees). En el cas que s'hagi detectat un tipus de dades no compatible amb les nostres llibreries, es mostrarà una icona d'alerta, i en cap cas podrà ser recuperada.

Existeix també, la columna *Data*, que ens indica amb una icona si la capa està carregada o no. Si s'intenta torna a carregar una capa carregada prèviament, o descarregar una capa no carregada, no s'efectuarà la càrrega de la capa que no compleixi aquestes regles.

Segons accedim a aquesta finestra de càrrega des de l'opció Nou Servidor, o Servidors, tindrà un comportament diferent. Si accedim des de Nou Servidor, una vegada recuperades les capes seleccionades, la finestra es tancarà, en canvi, si ho fem des de l'opció Servidors, no

ho farà, esperant que l'usuari emprengui altres accions com ara descarregar altres capes.
En el cas que s'indiqués descarregar una capa que ha estat modificada, se'ns demanaria si la volem salvar abans de descarregar-la o no.

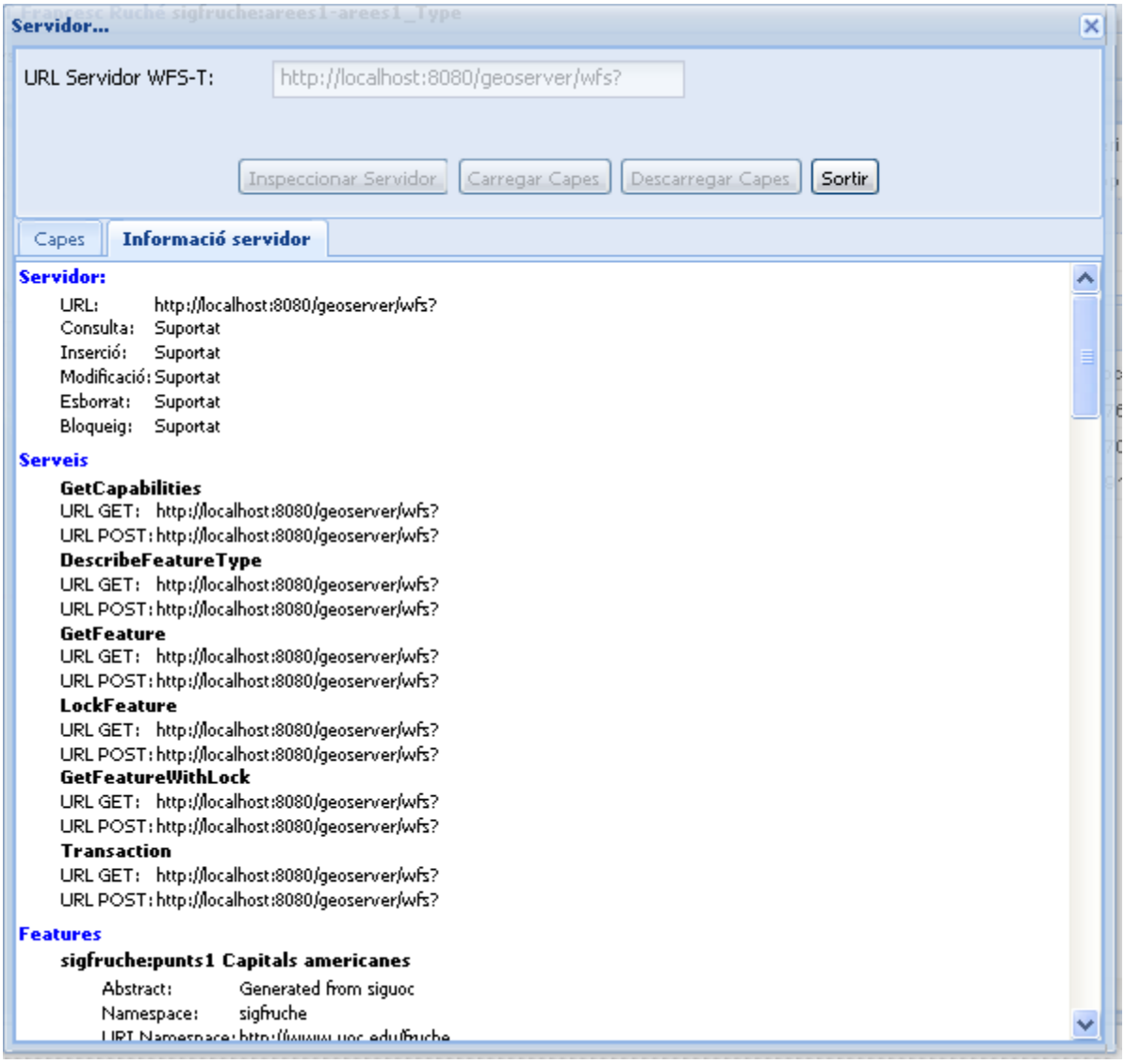


Figura 3.14 Finestra de càrrega – Informació de servidor.

També des d'aquesta finestra, es pot veure les característiques del servidor al que ens hem connectat, la seva adreça de servei base, les seves capacitats, els seus serveis amb les corresponents adreces i les característiques de cadascuna de les seves capes.

Explorador de capes i objectes geomètrics.
Aquest control, ens permetrà disposar d'una vista ordenada de totes les capes que tenim disponibles en un moment donat, així com una llista dels objectes geomètrics de la capa activa.

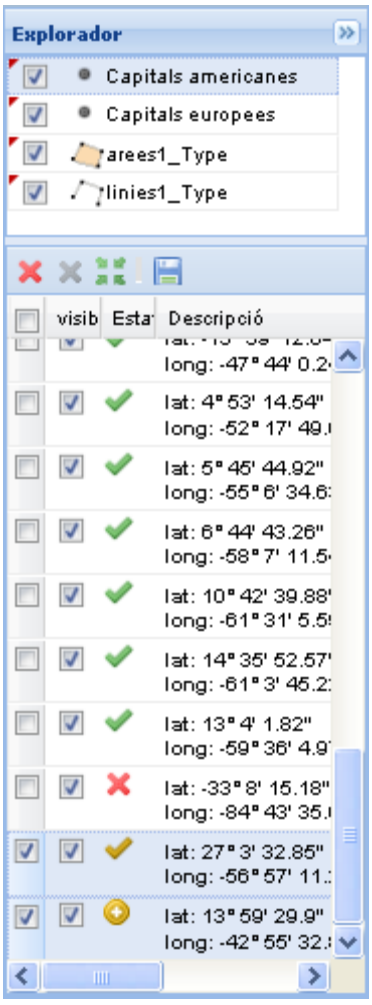


Figura 3.15 Explorador d'objectes

L'explorador de capes, ens indica totes les capes carregades, i quina és l'activa, que serà l'única seleccionada, ja que es tracta d'una llista de selecció simple.

Juntament amb la capa, se'ns indica el títol de la mateixa, i una icona representant el tipus de dada geomètrica que conté. Cada capa, disposa també, d'una caixa de selecció que farà que la capa sigui visible o no, utilitat que ens permetrà visualitzar algunes de les capes per poder-nos apropar als elements desitjats amb més comoditat

Cada vegada que seleccionem una capa, l'explorador d'objectes geomètric oferirà els objectes dels que la capa disposa.

L'explorador d'objectes geomètrics, ens ofereix una llista dels objectes que la capa seleccionada, conté. Aquesta llista és de selecció múltiple, per tant podrem tenir més d'un objecte selecciona i actuar sobre ells de forma conjunta.

De la mateixa forma que amb les capes, podrem amagar o visualitzar els objectes sobre el mapa que seleccionem amb la caixa de selecció de la columna 'visible'.
Per a cada objecte, s'ofereix una petita descripció que segons els tipus de dada geomètrica serà:

- **Punts**: latitud i longitud en graus, minuts i segons del punt
- **Línies**: la seva longitud en Quilòmetres o metres, segons la seva grandària.
- **Àrees**: La seva àrea en Quilòmetres quadrats.

La columna estat, ens ofereix informació amb icones sobre la situació actual de cada objecte, si l'objecte no ha estat modificat, si ha sofert variacions, si és un objecte nou, o si ha estat esborrat. Una vegada la capa hagi estat salvada, totes les icones passaran a l'estat de no modificat.

Figura 3.16 Icones d'estat

Una vegada algun objecte hagi estat modificat, s'activarà el botó de salvar la capa, a partir d'aquest moment, es pot salvar la capa activa. Aquesta acció és diferent de l'acció accessible des del menú, ja que aquella salvarà totes las capes carregades.

Per al conjunt d'objectes seleccionats, es poden realitzar les següents accions, encarà que només tindran efecte sobre els objectes que compleixin certes condicions:

- **Esborrar** els objectes seleccionats, si el seu estat és no esborrat.
- **Recuperar** els objectes esborrats, si es seu estat és esborrat, tornant a l'estat anterior. És a dir, si l'objecte estava en un estat modificat abans de ser esborrat, al recuperar l'objecte, tornarà a un estat modificat.
- **Centrar**: Fa que la interfície gràfica que representa el mapa es centri respecte als objectes seleccionats i faci un *zoom*, per tal de tenir-los tots dintre de la pantalla.

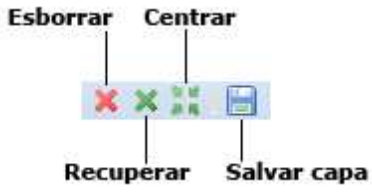


Figura 3.17 Botons d'acció

3.4.2. Interfície d'interacció geogràfica.

La interfície d'interacció geogràfica està construïda per la cartografia proporcionada per *Google Maps*, i controlada per les seves *API*'s. Sobre aquesta interfície, l'usuari pot interaccionar fent algunes de les accions que podria fer sobre la interfície d'interacció directa sobre les dades, i d'altres que són exclusives d'aquesta interfície.

Selecció de capa
L'usuari pot seleccionar una de les capes representades sobre el mapa per treballar amb ella, de la mateixa forma que ho podria fer seleccionant la capa amb l'explorador de capes.
L'usuari només haurà de situar el cursor sobre alguns dels objectes de la capa a seleccionar perquè aquesta s'activi canviant de color, seleccionant-la també als exploradors. De la mateixa forma una selecció a l'explorador de capes farà que els colors dels objectes sobre el mapa canviïn.

La situació de cada capa es mostra segons els següent codi de colors:

- **Blau**: la capa activa
- **Vermell:** la, o les capes inactives.

Selecció d'objectes
Podem seleccionar objectes de la capa activa fent un *Clic* a sobre, passant el color de l'objecte a verd. Una vegada seleccionat un objecte, aquest també es seleccionat a l'explorador d'objectes, i igual que amb les capes, una selecció sobre l'explorador fa que el color de l'objecte també canviï.

Moure objectes
Per modificar la situació d'un objecte, només podrem actuar sobre punt, o vèrtex de línies o àrees, iniciant una acció d'agafar i arrossegar, i finalitzar amb una acció de deixar anar, A mesura que es vagi movent l'objecte, s'anirà actualitzant la seva descripció (posició, longitud o àrea) a l'explorador d'objectes. L'objecte canviarà el seu estat a modificat, i ja no tornarà a l'estat de no modificat fins que no salvem la capa a la que pertany.

Afegir objectes
Per poder afegir objectes, ens caldrà començar amb un *Click*, i finalitzar en el cas de línies i àrees amb un *DblClik*. Si estem afegint un punt, només ens caldrà el *Click* inicial

S'ha de tenir en compte que no disposem d'un selector d'objecte a afegir, i que s'ha d'anar amb cura amb l'acció d'afegir, ja que el tipus d'objecte a afegir dependrà de la capa que tinguem activada en aquell moment. Si tenim una capa de línies activada, amb el *Click* inicial, s'afegirà una línia, i de la mateixa forma amb els punts i àrees.
El comportament adoptat a l'hora d'afegir objectes, és el mateix que proporciona l'API de *Google Maps* per defecte, per tant els usuaris tindran un comportament similar al que hagin pogut utilitzar amb anterioritat.

Esborrar punts i vèrtex
No es podran esborrar objectes des de la interfície d'interacció geogràfica, ja que aquesta

acció només es podrà fer des de l'explorador d'objectes. El que si es podrà fer és esborrar els vèrtex de línies i àrees efectuant un simple *Click* sobre ells.

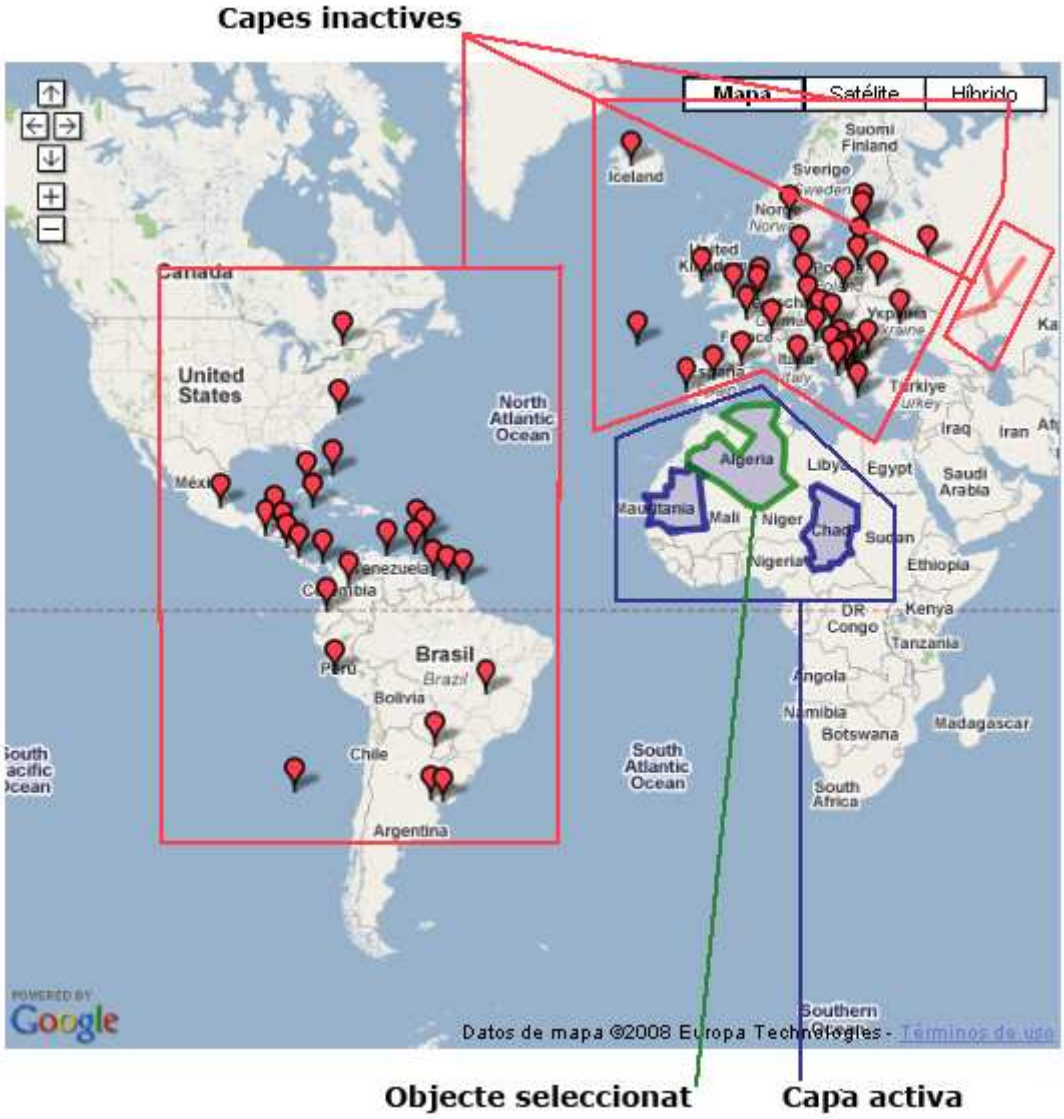


Figura 3.19 Selecció de capes i objectes

3.5. Controlador

La capa controladora s'encarregarà de connectar la vista amb el model, proporcionant un enllaç entre les accions que l'usuari efectua sobre l'IGU i les efectuades sobre el model.

És una forma de relació indirecta que té les seves avantatges, ja que aïlla la vista del model. Modificant el controlador, es pot modificar el comportament de la relació entre la vista i el model sense tenir que modificar aquestes.

Per exemple, si l'usuari modifica un element geomètric sobre el mapa, és el controlador qui se'n adona del canvi, transmetent-lo al model i modificant les dades que representen l'element modificat.

De la mateixa forma, l'usuari pot demanar la càrrega d'una capa del servidor, i serà el model l'encarregat de connectar-se al servidor demanat, recuperar les dades i dibuixar-les sobre el mapa.

Serà en aquestes classes que conformen el controlador, a on es farà ús de les *API's* de *Gogole-Maps*, associant objectes del model a objectes gràfics proporcionats pe les API's, i associant manegadors als events que aquests objectes gràfics poden llençar. D'aquesta forma un objecte geomètric del nostre model, tindrà associat un controlador que contindrà un objecte gràfic i que proporcionarà la resposta als events que aquest pugi oferir.

La idea, és associar objectes del model a objectes de l'API de *Google Maps* a través d'objectes controladors, que a més a més puguin reflectir els canvis a la interfície gràfica d'usuari.

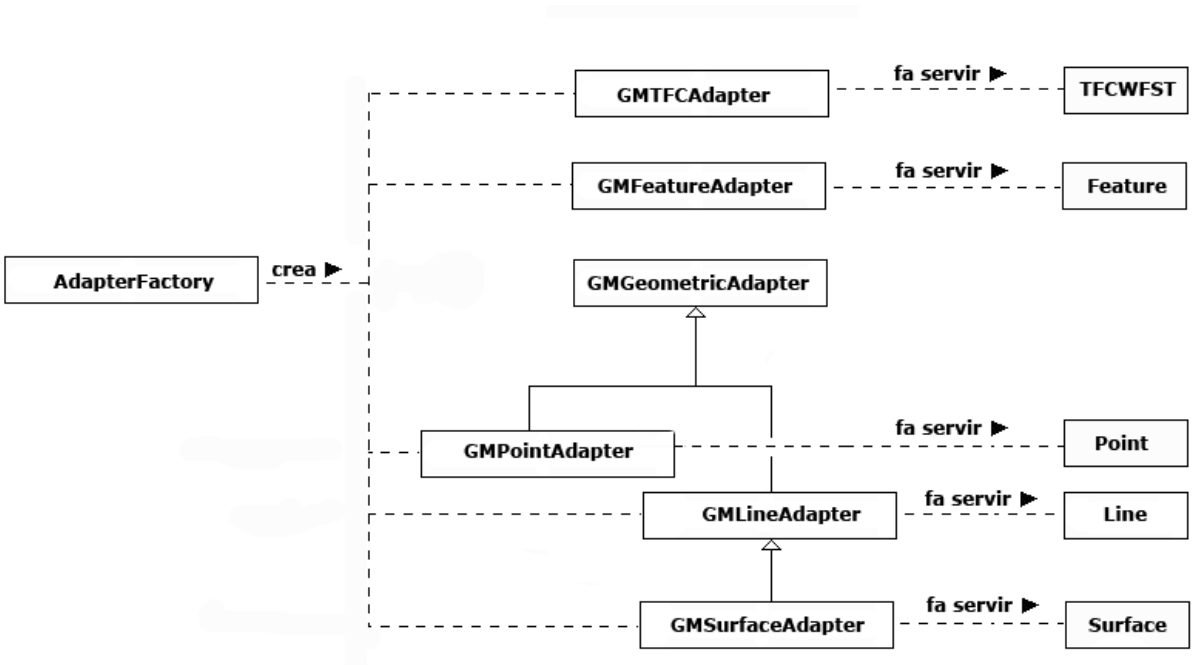
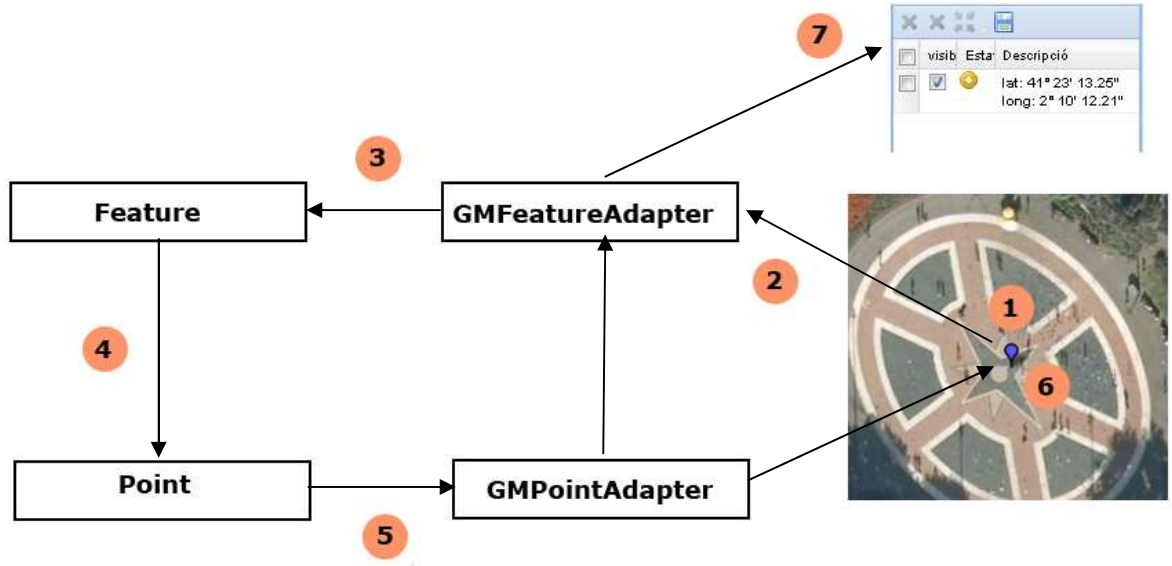


Figura 3.20 Diagrama UML estàtic de classes de la capa controladora

3.5.1. Exemple d'ús dels controladors.

Per il·lustrar la forma en que els controladors posen en contacte les capes de model i vista, explicarem en detall una situació en la que l'usuari afegeix directament un punt sobre el mapa.



1. L'usuari efectua un *Click* sobre una posició determinada al mapa, suposant que es té seleccionada una capa amb objectes geomètrics tipus punt.
2. El controlador de la capa activa, objecte de la classe *GMFeatureAdapter*, és l'encarregat d'atendre els events efectuats sobre el mapa, i detecta l'acció de l'usuari.
3. El controlador demana a l'objecte del model associat, una *Feature*, que afegeixi un punt amb una referència geogràfica en latitud i longitud.
4. La *Feature*, s'encarrega de crear un nou punt basada en la situació subministrada i demanar a l'*AdapterFactory* la creació d'un adaptador del tipus *GMPointAdapter* per tal d'associar-lo a l'objecte geomètric.
5. El nou adaptador associat, s'encarrega de crear un objecte gràfic *GMarker* i dibuixar-lo sobre al mapa, alhora que demana al controlador de la capa a la que pertany *GMFeatureAdapter* que actualitzi les vistes que cregui oportunes.
6. L'adaptador de la *Feature*, per últim, actualitza l'explorador d'objectes amb les dades del nou punt creat.

Per a totes les accions d'usuari sobre el mapa i sobre l'explorador, es desencadenaran un seguit de peticions entre la vista, el model i el controlador similars a la descrita

3.5.2. Classe de construcció d'adaptadors, *AdapterFactory*.

Per poder tenir un constructor universal de cadascuna de las classes del controlador, i que es puguin fabricar els objectes adequats per a les classes del model a les quals aniran associades, s'ha adaptat el patró *Factory* **[15]** a les nostres necessitats.

Aquesta classe té la missió de construir els adaptador necessaris en base a l'objecte que se li subministri, de tal forma que quan li demanem que construeixi un adaptador passant-li un objecte *Point* com a paràmetre, s'encarregarà d'esbrinar el tipus d'objecte rebut, i en base a la seva classe, construir l'adaptador adequat, *GMAdapterPoint* en aquest cas, realitzant l'associació entre l'objecte del model, l'adaptador creat i la interfície gràfica.

Per poder disposar d'un únic objecte d'aquest tipus al que demanarem la construcció d'adaptadors, s'ha emprat el patró de disseny *Singleton* **[12] [14]**, que a través d'un constructor privat i una referència a ell mateix, ens assegura la unicitat dels objectes de la classe.

Aquesta classe, disposa d'uns atributs a on s'indiquen les classes de l'objecte a crear en funció de l'objecte subministrat, cosa que ens permet modificar les classes associades simplement modificant aquests atributs, que d'altra banda podrien haver estat constants.

Aquest fet podria ser útil per a poder desenvolupar adaptadors específics amb altres comportaments que els dissenyats inicialment, simplement variant les referències a les classes a construir.

3.5.3. Adaptadors d'objectes geomètrics, *GMGeometricAdapter*, *GMPointAdapter*, *GMLineAdapter* i *GMSurfaceAdapter*.

Classe *GMGeometricAdapter*
Aquesta classe servirà de base per construir per herència els objectes geomètrics.
Proporcionant els atributs i mètodes comuns a tots ells, com podria ser l'estat del controlador, si es visible o no, si està seleccionat...
També proveeix mètodes amb una clara relació amb la vista, com per exemple mètodes per ocultar l'objecte gràfic, esborrar-lo, recuperar-lo...

Classe *GMPointAdapter*
La classe representa el controlador associat a un *Point* del model i un *GMarker* de l'API de *Google Maps*, encarregant-se de crear l'objecte gràfic a partir de l'objecte del model seguint les seves coordenades.

Donarà resposta als següents events generats a sobre del mapa:

- ***Drag***: Per iniciar un moviment del *GMarker*, marcar el *Point* com a modificat i demanar a l'*AdapterFeature* que modifiqui les coordenades de la vista a mesura que es va movent el punt.
- ***DragEnd***: Únicament demanarà a l'*AdapterFeature* que refresqui la vista.
- ***Click***: Marca l'objecte del model associat com a seleccionat, dibuixant-lo amb l'estat corresponent.
- ***MouseOver***: Té com a resposta el marcat de la capa a la que pertany l'objecte com a capa activa, si no ho és ja.

Un altre del mètodes més destacables serà el mètode *toHTML(),* que retornarà una descripció de la posició del punt amb la seva latitud i longitud expressada en graus minuts i segons.

Classe *GMLineAdapter*
La classe representa el controlador associat a una *Line* del model i un *GPolyline* de l'API de *Google Maps*, encarregant-se de crear-lo a partir de les coordenades dels vèrtex de l'objecte del model.

Els events als que donarà resposta seran els següents:

- ***Click***: Marca l'objecte del model associat com a seleccionat, dibuixant-lo amb l'estat corresponent.
- ***MouseOver***: Té com a resposta el marcat de la capa a la que pertany l'objecte com a capa activa, si no ho és ja.
- ***EndLine***: Finalitza la creació de la línea amb la que s'estava treballant.
- ***LineUpdated***: Cada vegada que es crea un vèrtex nou, o es modifica la posició d'un existent, és modifiquen tots els vèrtex de l'objecte del model associat i es marca com a modificat.

En aquesta classe el mètode *toHTLM()*, retornarà la longitud de la línia en metres o Quilòmetres, segons la seva grandària.

Classe *GMSurfaceAdapter*
El comportament d'aquesta classe, és molt similar al del *GMLineAdaper*, per tant és una extensió de la mateixa, i únicament variarà en el constructor de l'objecte gràfic que, en aquest cas serà un *GPolygon*.

L'únic mètode sobreescrit, serà el mètode *toHTML()*, que retornarà una descripció de l'àrea segons la seva superfície en Quilòmetres quadrats.

3.5.4. Adaptadors d'objectes de Servidor, *GMTFCAdapter* i *GMFeatureAdapter*.

Classe *GMTFCAdapter*
Aquest controlador, està associat a la IGU i a l'objecte arrel del model, i la seva única funció és la de tenir constància de quina de les capes ha estat seleccionada per l'usuari, per tal de fer les seleccions oportunes a l'IGU, dibuixar les capes amb els colors dels seus estats i actualitzar l'explorador.

Classe *GMFeatureAdapter*
El controlador de capes, serà el responsable d'escoltar el mapa per tal de detectar events de selecció i inserció d'objectes. Altres responsabilitats delegades en aquesta classe serien:

- Actualitzar la interfície gràfica d'interacció amb les dades.
- Demanar als controladors associats als objectes de la capa que es dibuixin segons l'estat de la capa.
- Respondre amb accions sobre el model a les accions de la interfície sota la seva responsabilitat.

Únicament donarà resposta a l'event Click del mapa, iniciant una acció diferent segons el tipus de capa activa de la següent forma:

- **Capa de Punts**: Afegirà directament un punt en la posició indicada.
- **Capa de Línies**: Crearà una *Polyline*, i la posarà en mode d'edició, amb la qual cosa es podran anant afegint vèrtex. Per tal d'evitar la creació de nous objectes en el procés d'edició de la línia nova, es des habilitarà temporalment l'event *Click* fins que finalitzi la creació.
- **Capa d'Àrees**: Exactament igual que amb la capa de línies, però creant un objecte *GPolygon*.

3.6. Estructura del programari

A continuació es descriuran els fitxers dissenyats i com estan distribuïts en directoris.

- **/** Index.html
  - **TFCWFST**
    - **core**
      - eventobject.js
      - hashtable.js
      - TFCWFST.js
      - WFSTComm.js
      - Server.js
      - Service.js
      - Feature.js
      - Geometry.js
      - SimplePoint.js
      - Point.js
      - Line.js
      - Surface.js
    - **IGU**
      - connexio.js
      - filemenu.js
      - gEntity.js
      - gFeatures.js
      - gmappanel.js
      - mainwin.js
      - **adapters**
        - AdapterFactory.js
        - GMFeatureAdapter.js
        - GMGeometricAdapter.js
        - GMLineAdapter.js
        - GMPointAdapter.js
        - GMSurfaceAdapter.js
        - GMTFCAdapter.js
      - **css**
        - IGU.css
      - **icons**
        - Icones dissenyades
      - **extjs**
        - Llibreries extjs

4. CONCLUSIONS

En aquest apartat es presenten les consideracions finals del Treball, un breu resum dels resultats obtinguts i les seves possibilitats d'ús real.

4.1. Resultats obtinguts

El resultat final del present Treball, ha estat prou satisfactori, sobre tot si tenim en compte la carència d'experiències prèvies tant en l'àrea dels SIG com en la programació sobre un llenguatge de guió com *Javascript*.

S'han assolit els objectius marcats, fent una gestió gràfica dels elements geomètrics geo referenciats proposats sobre una IGU tipus RIA. Igualment de profitosa ha estat la possibilitat de configurar tot un sistema SIG de persistència amb tots els elements necessaris com ara una base de dades i un servidor d'entitats geogràfiques.

Tot plegat ha constituït una experiència que pot ser un bon començament per aprofundir en un camp que avui en dia és en ple desenvolupament, ja que les iniciatives basades en serveis públic de cartografies estan canviant un món restringit fins ara a àmbits professionals.

S'ha pogut fer una recerca sobre els principals conceptes d'implementació, ja sigui amb l'orientació a objectes des de llenguatges de guió, fins a l'adopció de patrons de disseny, fet aquest, que ha estat un veritable descobriment, i que sens dubte ha col·laborat en poder portar a terme l'objectiu del Treball d'un forma molt més ordenada que la inicialment concebuda.

Respecte al producte final obtingut, hem de destacar l'esforç en proporcionar una interfície gràfica d'usuari el més moderna possible, dintre de les nostres possibilitats, que pogués satisfer usuaris avançats sense tenir que aprendre comportaments específics. Aquest fet, també ha estat una sorpresa, doncs amb la experiència disponible, mai s'havia imaginat el que amb *Javascript* es pot arribar a fer.

Les possibilitats reals d'ús de les llibreries dissenyades, passarien per poder disposar de servidors públics amb capacitats de persistència que es poguessin fer servir. Si aquests servidors es van constituint i consolidant a Internet, les llibreries poden tenir un ús real.

El fet és que també es pot fer servir les llibreries per a poder gestionar dades geomètriques en àmbits privats, ja que seria molt útil per la gestió de petits mapes amb informació especialitzada, com per exemple fer-lo servir per oferir informació des de llocs web sobre la situació de llocs d'interès turístic, o definir rutes de muntanya, o fins i tot fer petits jocs educatius per aprendre geografia.

4.2. Treball futurs

Les possibilitats de treballs futurs són molt grans, i en aquest treball, només s'ha fet la implementació bàsica per resoldre els problemes de primer nivell; connexió i representació de les dades.

A partir d'aquí, es podria fer tot un seguit de millores i evolucions que farien més útil el disseny implementat. Com a exemple podríem citar tot seguit unes quantes línies de millora:

- Millora de la vista:
  - Afinar el comportament implementat, donant-li una major lleugeresa.
  - Afegir funcionalitat, com per exemple; implementar cercadors d'objectes, adreces, llocs d'interès...
- Extensió dels controladors:
  - Construir controladors amb comportaments diferents que poguessin adaptar-se millor a certes tasques especialitzades, com per exemple; fent servir només el teclat, o una taula digitalitzadora.
  - Implementar nous controladors per altres serveis de cartografia que no fossin *Google-Maps*, per exemple construir controladors per *Yahoo Maps*.
- Evolució del model:
  - Es podria estendre la gestió a un conjunt més gran d'objectes geomètric, com per exemple ampliar el model per gestionar àrees complexes amb anells interiors i exteriors, o considerar la possibilitat d'incloure la altitud com a part del model.
  - Gestionar conjuntament amb els objectes geomètrics, altres tipus 'informacions com podrien ser la descripció del objectes, adreces relacionades, o imatges associades.

En resum, el ventall de millora i adaptació és molt ample, i només subjecte a la imaginació i a les hores de treball.

4.3. Possibilitats d'integració

Com hem comentat abans, existeixen possibilitats reals de fer servir les llibreries dissenyades per gestionar dades geomètriques de servidors públics o privats que permetessin unes grans possibilitats d'aplicació.

També és interessant les possibilitats de construcció a sobre de les nostres llibreries, doncs al separar el disseny per capes, es podria, a partir del model construir una capa de vista i controlador adaptada a unes necessitats concretes, amb l'avantatge que dóna partir d'una implementació que resol els problemes de connexió i representació d'objectes.

GLOSARI

AJAX *m* tècnica de desenvolupament web per a crear aplicacions amb interfícies elaborades.
en Asynchronous JavaScript And XML

API *f* Interfície de codi font que un component o biblioteca ofereix per a proveir altres aplicacions de funcionalitats i serveis.
en application program interface

aplicacions riques per Internet *f pl* Vegeu RIA**.**

application program interface *f* Vegeu API.

GML *m* Llenguatge basat en XML dissenyat per a modelar, transportar i emmagatzemar informació geogràfica.
en Geography Markup Language

HTTP *m* protocol de transport d'hipertext. Protocol que es sol utilitzar usat per recuperar pàgines web.
en HyperText Transfer Protocol.

mashup *m* Aplicació web híbrida que aprofita serveis de diferents proveïdors per integrar-los.

OGC *m* Organització que defineix els estàndards oberts i interoperables dins dels SIG. Persegueix acords entre les diferents empreses del sector que possibilitin la interoperació dels seus sistemes de geoprocés i facilitin l'intercanvi de la informació geogràfica en benefici dels usuaris.

Open Geospatial Consortium *m* Vegeu **OGC.**

RIA *f pl* Aplicacions web amb funcionalitat pròpia de les aplicacions d'escriptori.
en rich Internet applications

SOAP *m* protocol estàndard que defineix com dos objectes de diferents processos es poden comunicar entre ells intercanviant dades en XML
en Simple Object Access Protocol

XML *m* Llenguatge de marques ampliable, metallenguatge que permet definir gramàtiques de llenguatges específics.
en *Extensible Markup Language*

REFERÈNCIES

[1] John Snow. (2008), Wikipedia. 15 d'Octubre 2008.
http://es.wikipedia.org/wiki/John_Snow

[2] Canada Geographic Information System. (2008), Wikipedia 15 d'Octubre 2008.
http://en.wikipedia.org/wiki/Canada_Geographic_Information_System

[3] ESRI Canada. (2008) ESRI Canada Limited. 15 d'Octubre 2008.
http://www.esricanada.com/english/199.asp

[4] Google Maps API. (2008), Google Inc. 15 d'Octubre 2008.
<http://code.google.com/apis/maps/>

[5] Open Geospatial Consortium (OGC). Open Geospatial Consortium Inc. 15 d'Octubre 2008. <http://www.opengeospatial.org/>

[6] Apache Tomcat. The Apache Software Foundation. 15 d'Octubre 2008.
<http://tomcat.apache.org/>

[7] PostgreSQL. (2008). PostgreSQL Global Development Group . 15 d'Octubre 2008.
<http://www.postgresql.org/>

[8] PostGIS. (2008). Refractions Research. 15 d'Octubre 2008.
<http://postgis.refractions.net/>

[9] GeoServer. (2008). GeoServer. 15 d'Octubre 2008.
<http://geoserver.org>

[10] JavaScript: Mixing OOP and Event Driven Programming. (2008). Chris Pietschmann. 30 de Novembre 2008.
<http://pietschsoft.com/post/2008/11/JavaScript-Mixing-OOP-and-Event-Driven-Programming.aspx>

[11] JavaScript Implementation of Hashtable. (2005). Uzi Rafaeli. 30 de Novembre 2008.
<http://www.comet.co.il/en/articles/hash/article.html>

[12] The singleton design pattern in JavaScript . (2006). Kai Jäger. 30 de Novembre 2008.
<http://kaijaeger.com/articles/the-singleton-design-pattern-in-javascript.html>

[13] Modelo Vista Controlador. (2008). Wikipedia. 30 de Novembre 2008.
<http://es.wikipedia.org/wiki/Modelo_Vista_Controlador>

[14] Singleton. (2008). Wikipedia. 30 de Novembre 2008.
<http://es.wikipedia.org/wiki/Singleton>

[15] Factory Method (patrón de diseño). (2008). Wikipedia. 30 de Novembre 2008. <http://es.wikipedia.org/wiki/Factory_Method_(patr%C3%B3n_de_dise%C3%B1o)>

[16] Extjs. (2008).Extj, LLC. 30 de Novembre 2008. <http://extjs.com/>

BIBIOGRAFIA

Isabel Guitart Hormigo; Antoni Pérez Navarro (2007). Projectes, material docent. Barcelona: FUOC.

Alfons Bataller Díaz, Roser Beneito Montagut, Nita Sáenz Higueras, Rut Vidal Oltra (2008), Treball final de carrera, material docent. Barcelona FUOC.

Comas D., Ruiz E. (1993), Fundamentos de los Sistemas de Información Geográfica, Madrid: Ariel Geografia.

Domínguez Bravo, J. Breve (2000), Introducción a la Cartografía y a lo Sistemas de Información Geográfica (SIG). Madrid: Ed. CIEMAT.

Antoni Pérez Navarro, Albert Botella Plana, Anna Muñoz Bollas i altres (2008), Sistemes d'Informació Geogràfica i Geotelemàtica, Barcelona:FUOC

Danny Goodman (2001), JavaScript Bible Gold Edition, Nova York: Hungry Minds, Inc.

Michael Morrison (2008), Head First JavaScript, Sebastopol, California: O'Reilly Media, Inc.

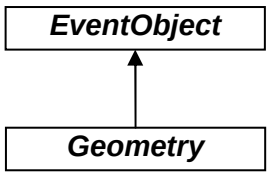
Graig Larman (2008), Applying UML and patterns, Edició lliure: Prentice Hall.

ANNEXOS

A API DEL MODEL

A.1 Classe *Geometry*

Herència



Atributs privats

- ***Id:String*** **[get,set]**
  Representa l'identificador únic de l'objecte, és un identificador que el servidor proporciona, i que en cap moment s'assigna des de la llibreria.
- ***feature:Feature*** **[get,set]**
  Representa la feature a la que pertany l'objecte.
- ***adapter:GeometryAdapter*** **[get,set]**
  Representa l'objecte adaptador necessari per als elements de vista.
- ***modified:boolean***
  Representa si l'objecte ha estat modificat o no.
- ***deleted:boolean***
  Informa sobre si l'objecte ha estat esborrat.
- ***newGeometric:boolean***
  Representa si l'objecte s'ha creat en temps d'execució, és a dir, no s'ha descarregat del servidor.

Mètodes públics

- ***isModified():boolean***
  Mètode de lectura de l'atribut ***modified***.
- ***setModified(value:boolean)***
  Mètode d'escriptura de l'atribut ***modified*** sense llençar cap event.
- ***modified()***
  Mètode d'escriptura de l'atribut ***modified*** llençant events.
  Llença l'event ***changeGeometry*****.**
- ***isDeleted():boolean***
  Mètode de lectura de l'atribut ***deleted***.
- ***delete()***
  Marca l'objecte com a esborrat.
  Llença els events ***changeGeometry*** i ***deletedGeometry***.
- ***unDelete()***
  Desmarca l'objecte com a esborrat, és a dir, el recupera.
  Llença els events ***changeGeometry*** i ***undeletedGeometry***.
- ***isNew():boolean***

Mètode de lectura de l'atribut ***newGeometric***.

- ***setNew(value)***
  Mètode d'escriptura de l'atribut ***newGeometric***.

Mètodes abstractes

- ***toString():String***
  Retorna una representació de l'objecte.

Events

- ***changeGeometry***
  Es llença cada vegada que es modifica l'objecte, passant-lo com a paràmetre, juntament amb la ***feature*** a la que pertany.
- ***deletedGeometry***
  Es llença cada vegada que s'esborra un objecte, passant-lo com a paràmetre, juntament amb la ***feature*** a la que pertany.
- ***undeletedGeometry***
  Es llença cada vegada que es recupera un objecte, passant-lo com a paràmetre, juntament amb la ***feature*** a la que pertany.

A.2 Classe *SimplePoint*

Atributs privats

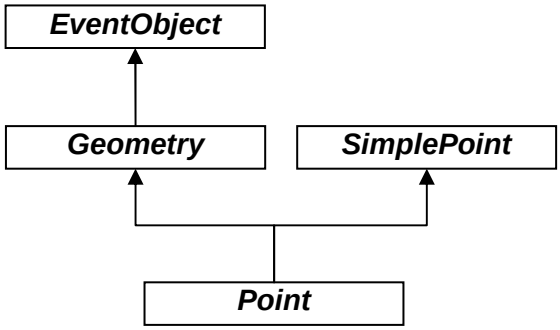
- ***lat:double*** **[get,set]**
  Representa la latitud
- ***lng:double*** **[get,set]**
  Representa la longitud

Constructor

- ***SimplePoint(lat:double , lng:double):Point***
  Construeix un objecte ***SimplePoint*** amb la posició proporcionada.

A.3 Classe *Point*

Herència



Constructor

- ***Point(feature:Feature, id:String , lat:double , lng:double):Point***
  Construeix un objecte ***Point*** amb l'identificador, la ***feature*** a la que pertany i la posició proporcionada.

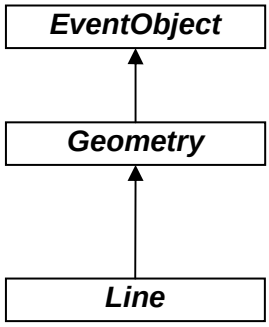
En aquest constructor es criden als constructors de les classes pare ***SimplePoint*** i ***Geometry***.

Mètodes públics

- ***setPosition(lat:double, lng:double):void***
  Canvia la posició del punt a la nova latitud i longitud, passant el punt a un estat de modificat.
  Si la posició del punt es modifica, llença l'event ***changeGeometry***
- ***toString():String***
  Retorna un *String* amb la cadena "Point:" seguit de l'***id***.

A.4 Classe *Line*

Herència



Atributs privats

- ***vertex:Array<Point>***
  Col·lecció de punts ordenats que representen els vèrtex d'una línia.

Constructor

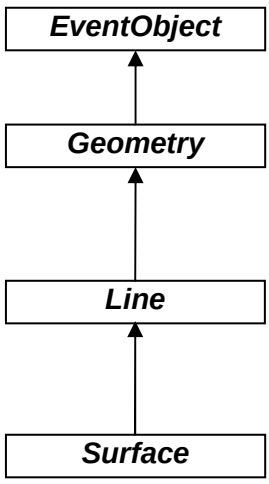
- ***Line(feature:Feature, id:String):Line***
  Construeix un objecte ***Line*** amb l'identificador i la feature a la que pertany.

Mètodes públics

- ***toString():String***
  Retorna un *String* amb la cadena "Line:" seguit de l'***id***.
- ***getPoints():Array<Point>***
  Retorna una matriu amb els vèrtex de la línia.
- ***addPoint(lat:double, lng:double)***
  Afegeix un punt a la línia. Serveix per a afegir els punts recuperats des del servidor WFS-T, ja que no llença cap event de modificació.
- ***updatePoints(points:Array<SimplePoints>)***
  Modifica els vèrtex de la línia als punts passats com a argument, modificant si s'escau aquests.
  Si es detecta una modificació, llença l'event ***changeGeometry.***
- ***clearPoints()***
  Elimina tots els vèrtex de la línia.

A.5 Classe Surface

Herència



Constructor

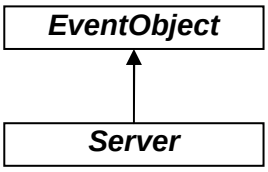
- **Surface(id:String , points:Array<Point>):Surface**
  Construeix un objecte ***Surface*** amb l'identificador i la feature a la que pertany.

Mètodes públics

- ***toString():String***
  Retorna un *String* amb la cadena "Surface:" seguit de l'***id***.

A.6 Classe *Server*

Herència



Atributs privats

- ***URLService:String*** **[get,set]**
  URL principal del servei *WFS-T*, adreça al que sempre ens haurem de dirigir per interrogar al servidor sobre les seves característiques i capacitats.
- ***query:boolean*** **[get,set]**
  Capacitat de consulta del servidor.
- ***Insert:boolean*** **[get,set]**
  Capacitat de d'inserció del servidor.
- ***update:boolean*** **[get,set]**
  Capacitat de modificació del servidor.
- ***del:boolean*** **[get,set]**
  Capacitat d'esborrat del servidor.

- ***lock:boolean*** **[get,set]**
  Capacitat de bloqueig del servidor.
- ***services:Hashtable<String,Service>*** **[get]**
  Taula indexada amb els serveis que ofereix el servidor.
- ***allFeatures:Array<Feature>*** **[get]**
  Array amb totes les capes de dades disponibles al servidor.
- ***features:Hashtable<String,Feature>*** **[get,set]**
  Taula indexada amb les capes de dades carregades en un instant donat.
- ***tfcwfst:TFCWFST***
  Objecte que representa el sistema.

Constructor

- ***Server(tfcwfst:TFCWFST)***
  Constructor que necessita l'objecte del sistema del qual dependrà.

Mètodes públics

- ***addService(service:Service)***
  Afegeix un *Service* a la taula de serveis
- ***getServiceByName(name:String):Service***
  Retorna el Service amb el nom demanat.
- ***getFeatureByName(name:String):Feature***
  Retorna la Feature amb el nom demanat.
- ***addFeature(feature:Feature)***
  Afegeix una ***Feature*** a l'*Array* de Features disponibles pel servidor.
- ***loadCapabilities():void***
  A través de la *URL* del servei proporcionada, es posa en contacte amb el servidor, i es configura les seves capacitats, els seus serveis i les features disponibles.
  Llença els events ***initLoadCapabilities*** i ***loadedCapabilities*****.**
- ***loadFeature(feature:Feature)***
  Carrega una Feature des del servidor, y la marca com a carregada
  Llença els events ***initLoadFeature*** i ***loadedFeature*****.**
- ***saveFeature(feature:Feature)***
  Salva la ***Feature*** indicada.
  Llença els events ***initSaveFeature*** i ***savedFeature*****.**

Events

- ***initLoadCapabilities***
  Es llença cada vegada que s'inicia la càrrega de capacitats del servidor.
- ***loadedCapabilities***
  Es llença cada vegada que es carrega les capacitats del servidor.
- ***initLoadedFeature***
  Es llença cada vegada que s'inicia la càrrega d'una ***Feature***.
- ***loadedFeature***
  Es llença cada vegada que es carrega una ***Feature***.
- ***initSaveFeature***
  Es llença cada vegada que s'inicia la guarda d'una ***Feature***.
- ***savedFeature***
  Es llença cada vegada que finalitza la guarda d'una ***Feature***.

A.7 Classe *Service*

Herència

Atributs privats

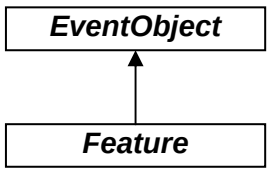
- ***name:String*** **[get]**
  Nom del servei amb el que volem contactar amb el servidor per a una tasca concreta.
  El seus possibles valors són:
  DescribeFeatureType
  GetFeature
  Transaction
- ***urlGET:String*** **[get]**
  URL d'accés al servei amb el mètode *HTTP GET.*
- ***urlPOST:String*** **[get]**
  URL d'accés al servei amb el mètode *HTTP GET*.

Constructor

- ***Service(name:String , urlGET:String, urlPOST:String): Service***
  Constructor amb els valors de tots els atributs.

A.8 Classe *Feature*

Herència



Atributs privats

- ***server:Server*** **[get]**
  Servidor al que pertany la ***Feature***.
- ***name:String*** **[get]**
  Nom de la feature, que sol ser l'espai de noms i la feature, per exemple "sigfruche:punts1".
- ***namespaceURI:String*** **[get]**
  URI de l'espai de noms emprat.
- ***title:String*** **[get]**
  Títol associat.
- ***abstract:String*** **[get]**
  Descripció detallada.

- ***defaultSRS:String*** **[get]**
  SRS per defecte emprat per la ***Feature***.
- ***outputgml3:boolean*** **[get]**
  Indica si la sortida es pot fer en GML3.
- ***namegeometric:String*** **[get,set]**
  Nom de l'atribut associat a la dada geomètrica principal
- ***typegeometric:String*** **[get,set]**
  Nom de la classe geomètrica principal.
- ***loadedData:boolean*** **[get,set]**
  Indica si s'han carregat les dades.
- ***valid:boolean*** **[get,set]**
  Indica si la ***Feature*** és vàlida per a ser tractada per la llibreria.
- ***adapter:FeatureAdapter*** **[get,set]**
  Adaptador associat a la ***Feature***.
- ***modified:boolean*** **[get,set]**
  Indica si cap dels elements geomètrics s'ha modificat, per tant la ***Feature*** queda pendent de ser salvada.
- ***geometrics:Array<Geometric>*** **[get]**
  Array que conté els elements geogràfics.

Constructor

- ***Feature(server:Server, name:String, namespaceURI:String, title:String, abstract:String, defaultSRS:String, outputgml3:String, namegeometric:String, typegeometric:String): Feature***
  Constructor amb els valors de tots els atributs inicials.

Mètodes públics

- ***toString():String***
  Retorna un descriptor composat per el nom i el títol.
- ***isPoint():boolean***
  Indica si la ***Feature*** conté punts.
- ***isLine():boolean***
  Indica si la ***Feature*** conté línies.
- ***isArea():boolean***
  Indica si la ***Feature*** conté àrees.
- ***getNamespace():String***
  Retorna l'espai de noms.
- ***addGeometric(geometric:Geometric)***
  Afegeix un element geomètric, sense considerar que és una modificació. Es fa servir per associar els elements geomètrics recuperats des del servidor..
- ***addNewGeometric(geometric:Geometric)***
  Afegeix un element geomètric, i dóna per modificada la ***Feature***.
  Llença l'event ***changeFeature*.**
- ***removeGeometric(index:integer)***
  Elimina un element geomètric pel número d'índex dintre de l'*Array*. No considera modificada la ***Feature***, ja que es fa servir en el procés d'actualització.
- ***modified(sender:Object ,scope:Object , msg:String)***

Manegador d'events per recollir les modificacions dels elements geomètrics dependents de la ***Feature.***
Llença l'event ***changeFeature*.**

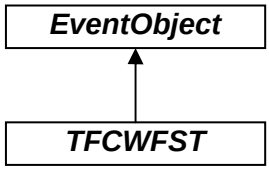
- ***destroyGeometrics()***
  Destrueix l'objecte i els elements geomètrics dependents, així com els adaptadors, si existeixen.

Events

- ***changeFeature***
  Es llença cada vegada que la ***Feature*** es modifica, indicant que s'ha de salvar.

A.9 Classe TFCWFST

Herència



Atributs privats

- ***comm:WFSTComm*** **[get]**
  Referència a un objecte de comunicacions.
- ***servers:Hashtable<String,Server>*** **[get]**
  Taula indexada amb tots els servers que s'hagin representat durant l'execució.
- ***adapter:Adapter*** **[get]**
  Adaptador associat a l'objecte
- ***lastURLServiceAdded:String***
  Darrera *URL* utilitzada, serveix per inicialitzar-la i conservar-la com a base per a futurs servidors.

Constructor

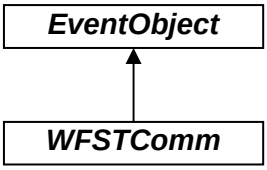
- ***TFCWFST(): TFCWFST***
  Constructor sense paràmetres.

Mètodes públics

- ***existServer(id:String)***
  Retorna si un servidor identificat per la seva URL ja ha estat carregat
- ***getServerById(id:String):Server***
  Retorna el servidor identificat pel paràmetre.
- ***addServer(server:Server)***
  Afegeix un servidor a la taula de servidors.
- ***getNewServer():Server***
  Crea un nou Servidor buit, per a poder configurar-lo.
- ***isModified():boolean***
  Retorna si alguna de les ***Feature*** d'aquest servidor ha estat modificada.
- ***loadAdapter(GUI:Object)***
  Carrega l'adaptador amb la referència d'una interfície gràfica d'usuari.

A.10 Classe *WFSTComm*

Herència



Constructor

- ***WFSTComm():WFSTComm***
  Constructor sense paràmetres.

Mètodes privats

- ***getRequest(url:String,xml:String):XML***
  Mètode que s'encarrega d'enviar una petició a una *URL* determinada amb un contingut i retornar el document rebut.
- ***createRequest():{XMLHttpRequest || ActiveXObject}***
  Mètode que crea un objecte de petició, ja sigui un ActiveXObject o un XMLHttpRequest, depenent del navegador utilitzar, per després poder fer les peticions necessàries.
- ***getCapabilitiesXML(server: Server) :XMLDocument***
  Recupera l'*XML* de la petició *getCapabilities*
- ***getDescribeFeatureTypeXML(server:Server ,feature:Feature) :XMLDocument***
  Recupera l'*XML* de la petició *DescribeFeatureType*
- ***getFeatureXML(server:Server ,feature:Feature) :XMLDocument***
  Recupera l'*XML* de la petició *GetFeature*
- ***parsePoint(feature:Feature ,xmlDoc:XMLDocument)***
  Parceja els elements ***Point*** d'un document *XML* i els afegeix a la ***Feature***.
- ***parseLine(feature:Feature ,xmlDoc:XMLDocument)***
  Parceja els elements ***Line*** d'un document *XML* i els afegeix a la ***Feature***.
- ***parseArea(feature:Feature ,xmlDoc:XMLDocument)***
  Parceja els elements ***Surface*** d'un document *XML* i els afegeix a la ***Feature***.
- ***getTransactionXML(feature:Feature):XMLDocument)***
  Recupera l'*XML* de la petició *Transaction*
- ***getXMLDelete (geo:Geometry ,feature:Feature) :String***
  Retorna l'*XML* d'un element si s'ha d'esborrar
- ***getXMLInsert (geo:Geometry ,feature:Feature) :String***
  Retorna l'*XML* d'un element si s'ha d'inserir
- ***getXMLUpdate (geo:Geometry ,feature:Feature) :String***
  Retorna l'*XML* d'un element si s'ha de modificar.
- ***geoToXML (geo:Geometry ,feature:Feature):String***
  Retorna l'*XML* d'un element

Mètodes públics

- ***loadCapabilities(server:Server)***

Recupera les capacitats d'un servidor

- ***getFeature(server:Server , feature:Feature)***
  Recupera les dades d'una ***Feature***
- ***getTransaction(feature:Feature)***
  Salva les dades modificades d'una ***Feature***

B API DEL CONTROLADOR

B.1 Classe *GMTFCAdapter*

Herència

GMTFCAdapter

Atributs privats

- ***tfcwfst:TFCWFST***
  Representa l'objecte principal del model, a través del que tindrem accés a tots els seus objectes.
- ***GUI:Object*** **[get]**
  Representa l'objecte principal de la interfície gràfica d'usuari.
- ***featureSelected:Feature***
  Feature seleccionada al IGU per l'usuari.

Mètodes públics

- ***select(feature:Feature)***
  Realitza la selecció a l'IGU de la Feature passada com a paràmetre.

B.2 Classe *GMFeatureAdapter*

Herència

GMFeatureAdapter

Constructor

- ***GMFeatureAdapter(feature:Feature,GUI:Object):GMFeatureAdapter***

Atributs privats

- ***feature:Feature***
  Feature associada a l'adaptador.
- ***GUI*** **[get]**
  GUI que l'adaptador controlarà.
- ***visible:boolean*** **[get]**
  Indica si la capa és visible o no.
- ***selected:boolean*** **[get]**
  Indica si la capa està seleccionada
- ***listener:GEventListener***
  Manegador del event *Click* sobre el mapa.
- ***overlaySelected:GOverlay***
  Overlay seleccionat a l'IGU
- ***indextodelete:integer***
  Indica el vèrtex marcat per a esborrar.

Mètodes públics

- ***destroy()***
  Destrueix l'objecte i desapunta les seves referències.
- ***showAll()***
  Dibuixa els objectes de la capa.
- ***hideAll()***
  Amaga tots els objectes de la capa.
- ***hideEntity(element:Geometry)***
  Amaga un element en concret de la capa.
- ***select()***
  Selecciona la capa com a capa activa.
- ***addPoint(overlay:GOverlay,point:GMarker)***
  Afegeix un punt al model que s'ha afegit al mapa.
- ***addLine(overlay:GOverlay,point:GMarker)***
  Afegeix una línia al model que s'ha afegit al mapa.
- ***addSurface(overlay:GOverlay,point:GMarker)***
  Afegeix una àrea al model que s'ha afegit al mapa.
- ***selectGroup()***
  Dibuixa la capa com a capa seleccionada.
- ***unselectGroup()***
  Dibuixa la capa com a capa no seleccionada.

B.3 Classe *GMGeometricAdapter*

Herència

GMGeometricAdapter

Constructor

- ***GMGeometricAdapter(geometric:Geometry,GUI:Object):GMGeometricAdapter***

Atributs privats

- ***geometric:Geometry***
  Element geomètric associada a l'adaptador.
- ***GUI:Object***
  GUI que l'adaptador controlarà.
- ***GMgeometric:GOverlay***
  Objecte de l'*API* de *Google Maps* associat a l'adaptador.
- ***visible:boolean*** **[get,set]**
  Indica si l'objecte gràfic és visible.
- ***selected:boolean*** **[get]**
  Indica si l'objecte gràfic és seleccionat.

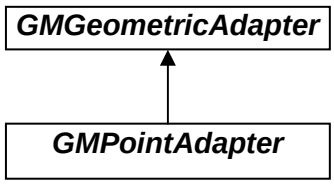
Mètodes públics

- ***hide()***
  Oculta l'objecte gràfic.

- ***delete()***
  Marca com a esborrat l'objecte geomètric del model.
- ***unDelete()***
  Marca com a no esborrat l'objecte geomètric del model.
- ***selectGeometric()***
  Selecciona l'objecte gràfic.
- ***selectGeometricByIGU(value:boolean)***
  El IGU ha selecciona l'objecte del model.
- ***selectGroup()***
  L'objecte gràfic forma part d'una capa seleccionada.
- ***unSelectGroup()***
  L'objecte gràfic no forma part d'una capa seleccionada.

B.4 Classe *GMPointAdapter*

Herència



Constructor

- ***GMPointAdapter(geometric:Geometry,GUI:Object):GMPointAdapter***

Mètodes públics

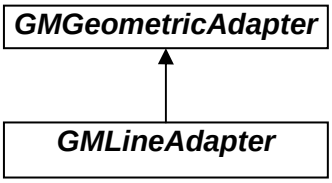
- ***show()***
  Mostra l'objecte gràfic.
- ***destroy()***
  Destrueix l'objecte i desapunta les seves referències.
- ***getPoints():Array<SimplePoint>***
  Retorna els punts que conté el Punt, en aquest cas un de sol.
- ***latInDegrees():String***
  Retorna la latitud en una representació de graus minuts i segons.
- ***toHTML():String***
  Retorna una Representació del punt en HTML
- ***lngInDegrees():String***
  Retorna la longitud en una representació de graus minuts i segons.
- ***toDegrees(v:double):String***
  Retorna la el paràmetre en una representació de graus minuts i segons.
- ***changePosition(latlng: LatLng)***
  Canvia la posició de l'objecte del model segons la posició d'un objecte gràfic.
- ***onDragEnd()***
  Manegador de l'event *DragEnd* de l'objecte gràfic.
- ***overPoint()***

Manegador de l'event *Over* de l'objecte gràfic.

- ***paintGMGeometric()***
  Dibuixa l'objecte gràfic segons l'estat de l'adaptador.

B.5 Classe *GMLineAdapter*

Herència



Constructor

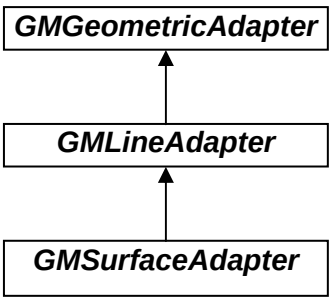
- ***GMLineAdapter(geometric:Geometry,GUI:Object):GMLineAdapter***

Mètodes públics

- ***show()***
  Mostra l'objecte gràfic.
- ***destroy()***
  Destrueix l'objecte i desapunta les seves referències.
- ***getPoints():Array<SimplePoint>***
  Retorna els punts que conté el Punt, en aquest cas un de sol.
- ***toHTML():String***
  Retorna una Representació del punt en HTML
- ***startDrawing()***
  Inicia l'edició d'un nou element
- ***onUpdated()***
  Es crida cada vegada que la línia es modificada
- ***addLineEnd()***
  Es crida al finalitzar l'edició d'una línia, si hi han menys de dos punts s'elimina la línia.
- ***updateLine()***
  Cada vegada que es modifica la línia, es modifiquen les dades del model.
- ***overLine()***
  Manegador de l'event *Over* de l'objecte gràfic.
- ***paintGMGeometric()***
  Dibuixa l'objecte gràfic segons l'estat de l'adaptador.

B.6 Classe *GMSurfaceAdapter*

Herència



Constructor

- ***GMSurfaceAdapter(geometric:Geometry,GUI:Object):GMSurfaceAdapter***

Mètodes públics

- ***show()***
  Mostra l'objecte gràfic.
- ***toHTML():String***
  Retorna una Representació del punt en HTML

B.7 Classe *AdapterFactory*

Herència

Constructor

- ***GMAdapterFactory():GMAdapter***

Atributs privats

- ***instance:AdapterFactory***
  Referència a l'únic objecte d'aquesta classe permès.
- ***adapterTFCWFSTClass:String***
  String que representa el nom de la classe per crear un *GMTFCAdapter*.
- ***adapterFeatureClass:String***
  String que representa el nom de la classe per crear un *GMFeatureAdapter*.
- ***adapterPointClass:String***
  String que representa el nom de la classe per crear un *GMPointAdapter*.
- ***adapterLineClass:String***
  String que representa el nom de la classe per crear un *GMLineAdapter*.
- ***adapterSurfaceClass:String***
  String que representa el nom de la classe per crear un *GMSurfaceAdapter*.

Mètodes públics

- ***getAdapter(obj: Object,GUI:Object)***
  Retorna un adaptador segons la classe de l'objecte del model passat com a paràmetre, creant-lo, i associant-lo l'objecte del model.