MareGIS

Documentación técnica do sistema

Documentación técnica dos sistema

Desenvolvemento interno	
Proxecto	MareGIS
Data	30/10/2011
Revisión	2

Datos de contacto

Quadralia, S.L.
NIF: B27742337
Rúa Tranvía nº 1, 1ª Planta
36400 O Porriño
Pontevedra

Tlfno: +34 986134376
+34 986481334
Fax: +34 986138050

info@quadralia.com
www.quadralia.com

Propiedade intelectual

Cofinanciación

Este proxecto foi cofinanciado por

XUNTA DE GALICIA
PRESIDENCIA
Secretaría Xeral de Modernización
e Innovación Tecnolóxica

FONDO EUROPEO
DE DESENVOLVEMENTTO
REXIONAL
"Unha maneira de facer Europa"
UNIÓN EUROPEA

Índice

1 Introdución

Neste documento recóllese unha descrición dos aspectos técnicos de proxecto MareGIS, así como unha visión exhaustiva das capas, clases e elementos que interveñen no correcto desenvolvemento da aplicación web.

Este manual está dirixido a programadores da solución GIS web.

2 Aspectos xerais

O deseño software do sistema está baseado no patrón de arquitectura “Modelo Vista Controlador (MVC)”, ofrecendo así una separación entre os datos da aplicación, a interface de usuario e a lóxica de negocio en tres compoñentes distintos.

3 Capa de Datos (Modelo)

A Capa de datos ofrece un sistema de comunicación entre a aplicación web e os datos almacenados na base de datos (neste caso un servidor de base de datos MySQL).

Os elementos almacenados na base de datos teñen a súa correspondencia coas clases empregadas na lóxica da capa de negocio, ofrecendo deste xeito, un nivel de abstracción para o resto da aplicación do sistema xestor de base de datos empregado.

As clases empregadas para a confección desta capa atópanse maioritariamente dentro da carpeta “classes”.

Nunha primeira visión desta carpeta observase unha clara diferenciación entre as clases empregadas para a confección das gráficas de datos e das clases empregadas no resto da lóxica da aplicación.

3.1 Capa de Datos: gráficas

IGrafica.php é a intereface que contén a declaración das funcións que deben implementar as gráficas para o correcto desenvolvemento desta. Por orden de invocación (e polo tanto de proceso lóxico) estas son a funcións que teñen definidas todas as gráficas:

- ***getDeclaration***(): Obtén a declaración en código JavaScript do tipo de gráfica a representar (barras, áreas, columnas, etc) e do aspecto visual empregado (títulos, subtítulos, tooltips, etc).
- ***getProcess***(): Devolve o código JavaScript de como serán tratados os datos, obtidos mediante a chamada AJAX correspondente, da gráfica desexada e do xeito en que estes serán incorporados á gráfica.
- ***GetMinimunFields***(): Esta función é a encargada de definir que campos da zona de parámetros son necesarios cubrir obrigatoriamente para poder xerar a gráfica. Cabe resaltar que esta función só devolve o Id dos campos HTML, delegando nos JavaScripts correspondentes da propia páxina os mecanismos de validación.
- ***GetExplanation***(): Obtén un texto explicativo da gráfica que se está a xerar.
- ***getHiddenFields***(): De maneira análoga á función anterior, obtéñense os identificadores dos campos da zona de parámetros que é preciso ocultar.
- **getGraph**(IdunidadGestion, idZona, idEspecie, fechaInicio, fechaFin, idArte, idEmbarcacion, idTipoTalla): Función que se invoca de xeito estático mediante unha chamada AJAX, e que nos devolve, en función dos parámetros que se establecen, os datos que se representarán.

A maiores destas funcións, tódalas gráficas teñen unha estrutura similar para un correcto entendemento e depuración da obtención e tratamento de datos que implica a función **getGraph**. Por orden de invocación son os seguintes:

- ***getData***(*IdunidadGestion, idZona, idEspecie, fechaInicio, fechaFin, idArte, idEmbarcacion*): Función encargada de realizar a chamada SQL correspondente cos parámetros establecidos e devolver un array cos resultados obtidos.
- ***procesarDatos***(*data*): Recibe como parámetro o conxunto de resultados da base de datos obtidos anteriormente e procésaos, de ser necesarios, para obter os resultados esperados por a lóxica da gráfica (agrupacións, sumatorios, etc).
- ***exportData***(*data*): Transforma os datos procesados en estruturas de datos, xeralmente arrays, que sexan fáciles de tratar polo JavaScript correspondente (función getProcess).

Por último cabe destacar o arquivo Grafica.php, que é o arquivo que se invoca mediante AJAX e que serve para redirixir o fluxo da aplicación, invocando a clase correspondente para a xeración da gráfica, en función do parámetro "type" que chega por POST.

3.2 Capa de Datos: lóxica de negocio

Debido á natureza da aplicación creáronse as seguintes clases para ofrecer un correcto desenrolo (descritas dende as máis xenéricas ás máis concretas).

- **cUnidadGestion** (arquivo "unidadGestion.php"): Representa unha reserva (ou conxunto de zonas xeográficas) baixo as que monitorizar capturas e mostras.
- **cZonaPesca** (arquivo "cZonaPesca.php"): Cada unha das zonas nas que se divide unha unidade de xestión.
- **cUnidadGestionZona** (arquivo "rUnidadGestionZona"): Establece a relación entre unidades de xestión e zonas.
- **cCoordenada** (arquivo "cCoordenada"): Representa cada un dos pares de coordenadas xeográficas (lonxitude, latitude) que delimitan unha zona dunha unidade de xestión.
- **cUsuario** (arquivo "usuario.php"): Representa un usuario da aplicación web. A súa principal funcionalidade é obter información do usuario conectado, así como dos permisos que ten asociados ao seu perfil.
- **cPermiso** (arquivo "permiso.php"): Representa cada un dos diferentes perfiles que se poden aplicar a un usuario.
- **cUsuarioPermiso** (arquivo "PermisoUsuario.php"): Establece a relación entre un usuario e todos os permisos que se lle aplican en cada unha das unidades de xestión establecidas.
- **cPuntosEstaticos**: (arquivo "PuntosEstaticos.php"): Representa cada un dos puntos que se mostran nas diferentes capas dos mapas do servidor GIS.
- **cTipoPunto** (arquivo "TipoPunto.php"): Cada unha das categorías nas que se clasifica un punto estático.

Todas as clases siguen unha estrutura común, personalizada en función dos atributos existentes na natureza de cada entidade:

- Declaracións e propiedades: Representa os atributos que teñen cada identidade representada. As declaracións son variables privadas para almacenar dita información, que se fan visibles a través das propiedades (funcións gets e sets).
- Métodos privados: Conxunto de funcionalidades que son administradas pola propia clase para o seu correcto funcionamento:
  - ***ObtenerParametros***(*IncluirId*): Establece todos os parámetros para as chamadas SQL da clases. Como parámetro especifícase se se inclúe o valor do identificador ou non.
  - ***Cargar***(*Instancia*, *Fila*) : Carga un rexistro da base de datos nunha instancia da clase.
  - ***Actualizar***(): Actualiza un rexistro existente na base de datos.
  - ***Insertar***(): Garda un novo rexistro na base de datos e, posteriormente, obtén o seu identificador.
  - ***GuardarDependencias***(): Garda calquera dato dependente do rexistro (clases dependentes, táboas relacionadas, etc).
- Métodos públicos: funcionalidades expostas que poden ser invocadas:
  - ***Eliminar***(): Elimina o rexistro actual da base de datos.
  - ***Guardar***(): Garda o rexistro na base de datos. Baseándose en se está establecido un identificador ou non, invocarase a función Insertar ou Actualizar.
  - ***CargarPorId***(*IdRegistro*): Carga unha instancia da clase en función do seu identificador.
  - ***ObtenerTodos***(): Obtén un array con instancias de todos os rexistros da base de datos correspondente.

Mención especial require a clase "**BaseDatos**", localizada no arquivo "dbManager.php" dentro da carpeta "utils", que é a encargada de xestionar as conexións coa base de datos, realizar as consultas e devolver os resultados, co fin de proporcionar un maior nivel de abstracción da aplicación co sistema xestor de base de datos. Todas as peticións SQL pasan a través desta clase.

Para un correcto funcionamento, esta clase precisa ter acceso á clase Conf (arquivo "configuration.php" dentro da carpeta "config"), onde se atopan os parámetros de conexión necesarios para establecer a comunicación coa base de datos.

4 Capa de negocio (Controladores)

Implementada na súa maior parte en código JavaScript e PHP, esta capa encárgase de recoller as peticións do usuario e ofrecerlles a resposta esperada. Para lograr isto empréganse chamadas en asíncronas en AJAX e envío de formularios a páxinas PHP.

Como punto fundamental desta capa encóntrase o arquivo index.php, que é o encargado de proporcionar a navegabilidade dentro da aplicación web en función do parámetro "action" pasado na URL de petición. O arquivo cargará entón os controladores (scripts) necesarios e as plantillas visuais correspondentes en cada momento.

Ademais, podemos atopar a maior parte dos arquivos desta capa dento da carpeta "scripts", onde ademais das librerías de jquery, jqueryUI (User Interface), Highcharts (xeración de gráficas) e Openlayers, atópanse todos os procesos que se executan na máquina do usuario mediante JavaScript, e nas carpetas "controller" e "actions", onde se recollen os arquivos de lóxica de negocio a executar no servidor PHP.

Dentro dos arquivos JavaScript que se executan no equipo do cliente, é necesario destacar:

- **login.js**: Controla a chamada AJAX ao proceso de autenticación (arquivo "controller/login.php"), onde se comproba na capa de datos se existe o usuario co nome especificado e co contrasinal escrito, así como os permisos dos que dispón. Se os datos de identificación son correctos, outórgaselle acceso ao sistema.
- Quadralia.js: proporciona a interactividade cos mapas do sistema GIS, engadindo, amosando ou agochando as capas de puntos estáticos que se tratan no sistema.
- **graph.js** (carpeta "graph"): Recolle as funcións de refresco de datos e interacción coas gráficas. Está formado polas seguintes funcións:
  - ***initGraph***(*language*): Inicializa o aspecto da gráfica, así como o idioma que se empregará na súa visualización.
  - ***getTheme***(): Obtén o tema visual que se aplica a aspectos comúns de tódalas gráficas.
  - ***RefrescarDatos***(): Fai a chamada ao arquivo "classes/Graficas/Grafica.php" descrito anteriormente, establecendo os parámetros fixados por o usuario. Unha vez obtida a resposta, procesa o resultado e mostra ou oculta a gráfica en función de se existen datos cos filtros actuais.
  - ***getGraphType***() : Analiza a URL de chamada para distinguir o tipo de gráfica solicitada e axuntala na chamada AJAX.
  - ***hideGraph***(): Oculta de xeito visual a gráfica ao usuario.
  - ***showGraph***(): Mostra a gráfica ao usuario.
  - ***resetZoom***: Nas gráficas que o soporten, restablece o zoom ao seu nivel por defecto.
  - ***setChart***(*nivelDesglose*, *categories*, *data*, *series*, *colorSerie*, *nombreSerie*, *ejeY*): Permite facer gráficas de diferentes niveis establecendo novamente

todos os elementos que se utilizan para a súa confección.

- ***extraPieChart***(): Aplica diferentes melloras aos gráficos baseados en torta.

- **GraphForm.js** (carpeta “graph”) e **toolsForm.js** (carpeta “script”): Proporcionan rutinas de inicialización de formularios e de interacción con frames externos.

No tocante aos arquivos que se executan do lado do servidor, e que podemos facilmente localizar na carpeta “controller”, destacaremos:

- **login.php, logout.php** e **acceso.php**: Encargados da autenticación e peche de sesión dos usuarios do sistema. Unha vez verificada a identidade dun usuario, gárdanse os seus datos nunha variable de sesión, quedando deste xeito autenticado dentro do período de tempo configurado previamente no servidor, ou hasta que o usuario decida pechar a sesión, momento no cal se eliminarán os seus datos da variable de sesión. Cada vez que sexa necesario comprobarase o estado do usuario invocando o arquivo acceso.php.
- **Xmltranslator.php** e **cambiodeidioma.php**: Encargadas de tratar cos arquivos de localización que se atopan dentro da carpeta “locale”. Proporcionan a tradución de termos e de cambio de idioma en tempo real da aplicación.
- Carpetas “**ferramentas**”, “**graficas**” e “**informacion**”: Conteñen unicamente as chamadas aos arquivos JavaScript a incluír en cada vista.

Por último temos a carpeta “actions”, onde se recollen as accións que realiza o usuario dende a parte de ferramentas e que serven como comunicación coa capa de acceso a datos:

- **addPunto.php**: Engade un novo punto a unha zona previamente seleccionada.
- **addZona.php**: Engade unha nova zona a unha unidade de xestión previamente seleccionada.
- **deleteZona.php**: Elimina unha zona previamente seleccionada.
- **modifyZona.php**: Modifica os datos dunha zona previamente seleccionada.
- **addUnidadGestion.php**: Engade unha nova unidade de xestión ao sistema.
- **deleteUnidadGestion.php**: Elimina unha unidade de xestión previamente seleccionada.
- **modifyUnidadGestion.php**: Modifica os datos dunha unidade de xestión previamente seleccionada.

4.1 Arquivos auxiliares

Aparte dos arquivos vistos no apartado anterior, existen unha serie de clases e funcións que axudan ao desenvolvemento da aplicación. Ditos arquivos conteñen funcionalidades comúns durante toda a execución da aplicación:

- **utils.php** (carpeta “utils”): Formado polas seguintes funcións:
  - ***getsha256***(*phrase*): Obtén a cadea resultante despois de aplicarlle o parámetros de entrada o algoritmo criptográfico SHA-2 (SHA-256).
  - ***t_***(*SourceString*, *lang*): Traduce a cadea de texto especificada no parámetro “SourceString” ao idioma especificado no parámetro “lang”.
  - ***tgen_***(*SourceString*): Traduce a cadea de texto especificada no parámetro “SourceString” ao idioma establecido durante a execución da aplicación web.
  - ***encrypt***(*string*): Cifra a cadea especificada nun sistema de criptografía reversible.
  - ***decrypt***(*string*): Descifra unha cadea cifrada previamente mediante a función encrypt().
  - ***showError***(*exc*): Mostra información do erro especificado como parámetro de entrada.
  - ***to_utf8***(*in*): Verifica e cambia, en caso de ser necesario, o xogo de caracteres a UTF-8 do parámetro de entrada.
  - ***StartsWith***(*cadena, textoABuscar, caseSensitive*): Verifica se a cadea especificada como primeiro parámetro comeza polo texto especificado como segundo parámetro. Ademais, pódese forzar unha comparación sensible a maiúsculas e minúsculas establecendo a verdadeiro o último parámetro de entrada, que por defecto toma valor “falso”.
  - ***checkReferer***() e ***checkRefererAndRedirect***(): Funcións de seguridade que verifican as chamadas directas a arquivos da aplicación sen pasar pola lóxica desta. Podemos optar simplemente por verificar, ou por verificar e devolver á páxina principal en caso de non cumprir a verificación.
- **Combos.php** (carpeta “utils”): Obtén todos os elementos da capa de datos que teñen que mostrar as diferentes listas de selección dos formularios.
- **configuration.php** (carpeta “config”): Arquivo onde se recollen tódolos parámetros de configuración do sistema, como os relativos á conexión á base de datos, ou os relativos aos modos de depuración.

5 Capa de presentación (Vistas)

Encargada de presentar todo o sistema ao usuario baixo unha interface amigable e ofrecerlle mecanismos de comunicación con el. Programada maioritariamente baixo código HTML e CSS, pódense localizar os arquivos correspondentes nas carpetas “template”, onde se ofrece un esquema da estrutura da páxina (títulos, bloque, cabeceiras, etc) e “blocks”, onde se recolle información detallada dos elementos que compoñen cada un dos bloques especificados no arquivo da plantilla correspondente. A carga das vistas é controlada en todo momento polo arquivo index.php.

A maiores, emprégase o arquivo de JavaScript “menu.js” (carpeta “script”) para mostrar ao usuario un menú despregable cos diferentes tipos de gráficas dispoñibles.

6 Outra documentación

O sistema apóiase de diferentes librerías externas para unha maior eficacia. A continuación proporciónase acceso a documentación destas librerías dende os seus sitios oficiais:

- Jquery: http://docs.jquery.com/
- Jquery UI: http://docs.jquery.com/UI
- Highcharts: http://www.highcharts.com/ref/
- OpenLayers: http://trac.osgeo.org/openlayers/wiki/Documentation
- PHP: http://php.net/docs.php
- MySQL: http://dev.mysql.com/doc/
- Zend Framework: http://framework.zend.com/manual/