SATODE

Diseño del Sistema e Implementación

Docentes-Tutores:
Sandro Moscatelli
Libertad Tansini

Estudiantes:
Fernando Carriquiry
Gabriel Fernández

1. Introducción 3
1.1. Propósito 3
1.2. Alcance 3
2. Consideraciones generales de diseño 4
3. Arquitectura 5
3.1.1. Base de datos 6
3.1.2. Despliegue de la aplicación 6
4. Modelo de dominio 10
5. Módulos 12
5.1. Clase de diseño modulo usuarios 13
5.2. Clase de diseño modulo registro de eventos 14
5.3. Clase de diseño modulo desastres 15
5.4. Clase de diseño modulo suministros 16
5.5. Clase de diseño modulo punto referencia 17
5.6. Clase de diseño modulo registro propiedades siniestradas 18
5.7. Clase de diseño modulo necesidades 22
6. Implementación 24
6.1. Herramientas 24
6.1.1. Lenguaje de programación – Java 24
6.1.2. Entorno de desarrollo - Eclipse 3.7 Indigo 24
6.1.3. Desarrollo de interfase de usuarios – GWT 24
6.1.4. Servicios de negocio - Spring Business Service 24
6.1.5. Persistencia – Hibernate 24
6.1.6. Reportes – Jasper 25
6.1.7. Control de versiones – Subversion 25
6.1.8. Base de datos - MySQL 25
6.2. Tecnología aplicada a la arquitectura 25
6.3. Planificación 26
6.4. Patrones y estándares 27
6.5. Procedimientos 29
7. Referencias 30
8. Anexos 32
8.1. Alcance del Sistema 32
8.1.1. Introducción 32
8.1.2. Propósito 32
8.1.3. Alcance 32
8.1.4. Planificación para lograr el alcance 32

1. Introducción

En este documento se presenta las distintas vistas del diseño del sistema, se exhiben las herramientas a utilizar, arquitectura y el modelo de dominio.

1.1. Propósito

El principal propósito del documento es registrar todos los aspectos relevantes del diseño de la aplicación, para el desarrollo del prototipo, para futuros desarrollos y modificaciones del sistema, documentar todas las herramientas utilizadas, patrones y estándares seguidos.

1.2. Alcance

El diseño pretende abarcar todas las funcionalidades descriptas en el documento alcance del sistema, que se encuentra como anexo a este documento.

2. Consideraciones generales de diseño

Para el diseño de los distintos artefactos del software consideramos importantes las propiedades no solo funcionales, sino también las propiedades inherentes al producto, como ser las sugeridas por los criterios GRASP[1].

Las principales propiedades que son deseables dada la realidad relevada son las siguientes:

Usabilidad

El sistema debe ser de fácil uso por parte de usuarios inexpertos, todos los casos de uso deben ser similares, de forma que el usuario pueda intuir la forma de interactuar con el sistema. Los aspectos visuales y estéticos influyen en la confianza y la credibilidad del sistema.

Extensibilidad

El sistema debe ser extensible, esto quiere decir que fácilmente se podrá agregar nuevos módulos, reutilizando la mayor cantidad posible de la funcionalidad ya desarrollada. Esta propiedad es importante para el sistema ya que el prototipo implementado puede ser extendido en trabajos futuros.

Mantenibilidad

El sistema debe ser fácil de monitorear y de *debuguear*, ya que son propiedades necesarias a la hora de solucionar problemas, debe tener bajo acoplamiento y alta cohesión en todos sus componentes, asegurando su fácil modificación y corrección.

Disponibilidad

Debido al uso distribuido del sistema en todo el país, es necesaria su alta disponibilidad desde los distintos centros de acción, por medio de un canal de comunicación, ya sea internet o una intranet. El sistema debe estar siempre en línea y accesible desde todo el país, ya que la gestión de un desastre se centraliza en la información que el sistema provee.

La aplicación debe contar con una mínima seguridad, pero no es el foco del problema, este requerimiento no condiciona la arquitectura, ni el despliegue de la aplicación, pero por temas de disponibilidad es deseable que se acceda por VPN [2], aunque no es un requisito indispensable. También es deseable comunicación HTTPS [3] para asegurar integridad de punta a punta y que los datos no sean interceptados o alterados en el medio de la comunicación.

En cuanto a confiabilidad, si bien es deseable un buen nivel de confiabilidad, no contamos con la descripción de los servidores utilizados para el despliegue de la aplicación, tampoco las condiciones y velocidades de las líneas de comunicación, ni un promedio de futuros usuarios concurrentes en el sistema. Por ello no podemos diseñar pruebas que aseguren una alta confiabilidad, pero si debe tener en cuenta no limitar el sistema de forma tal que sea inviable una alta confiabilidad.

En cuanto a eficiencia, consideramos que la aplicación se desplegará en uno o dos servidores a lo sumo y tomando en cuenta que el mayor uso de recursos que realizará el sistema será de memoria RAM y disco duro, no enfocaremos la solución al logro de una mayor eficiencia en cuanto al uso de recursos debido que estos son de bajo costo.

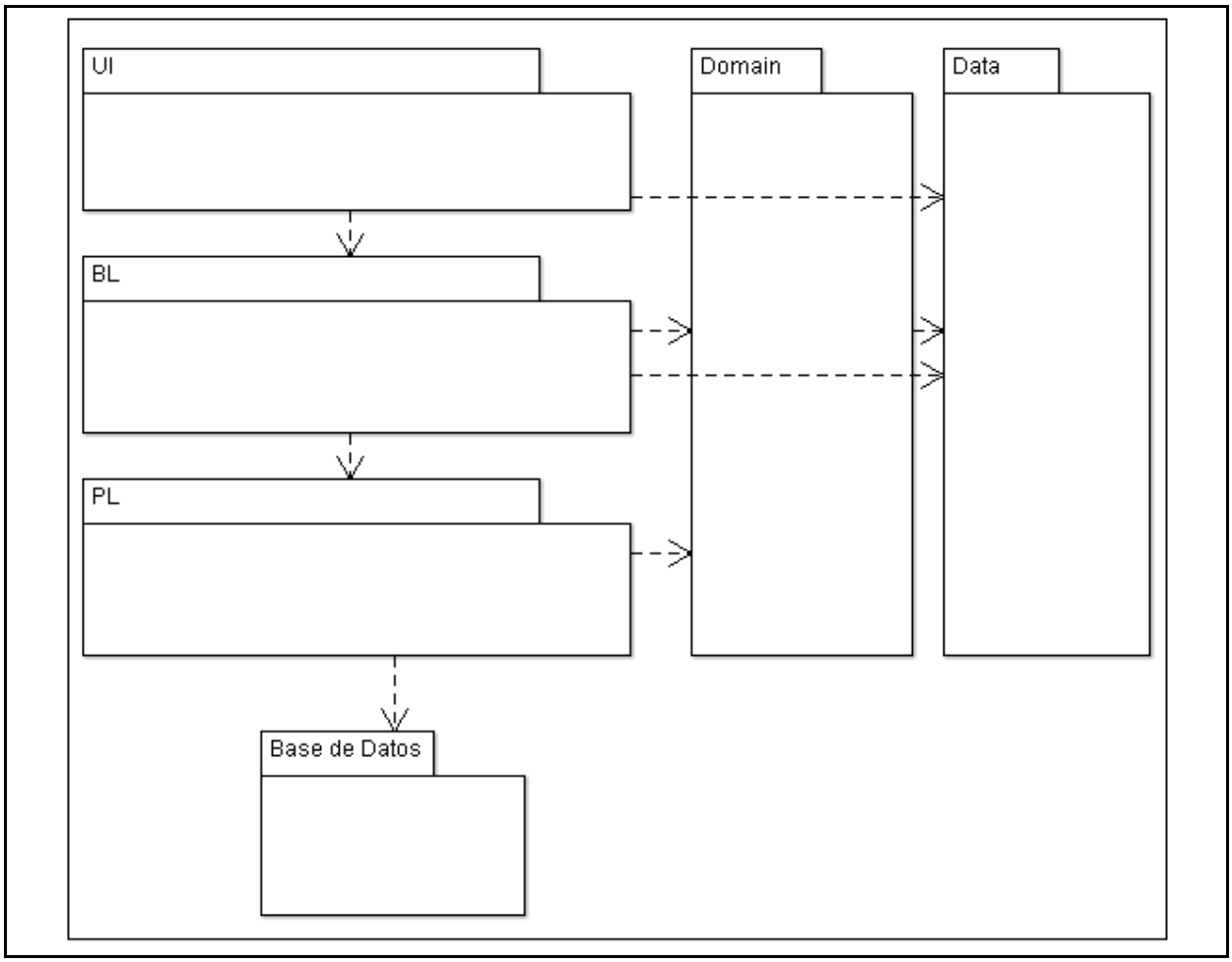
3. Arquitectura

En esta sección se presenta un diagrama que describe la arquitectura en forma general y se describe cada capa.

Los usuarios están distribuidos en todo el país y existe una gran diversidad de software de base en los equipos que utilizan para conectarse al sistema. Por lo que un sistema web se ajusta a dichos requerimientos ya que con solo contar con un navegador web corriendo en cualquier plataforma se accede al sistema. De esta forma actualizar el sistema tiene un costo muy bajo, comparado con una solución de escritorio, que debe ser instalada en cada pc y desarrollada para las distintas plataformas.

Se descarta arquitectura en dos capas ya que o bien la lógica de negocio se acopla a la presentación o a la persistencia. Ambos casos violan los criterios generales de extensibilidad debido al alto acoplamiento y al de mantenibilidad por la baja cohesión. También el alto acoplamiento genera mayores costos de mantenimiento y nuevos desarrollos, por la baja reutilización de los componentes.

La arquitectura escogida es la clásica arquitectura en tres capas ya que cumple con las consideraciones generales de bajo acoplamiento y alta cohesión en sus componentes, de esta forma se crean componentes que se pueden reutilizar o sustituir con un bajo impacto en el resto del sistema. Las capas son, interface de usuarios web, lógica y persistencia, se agregaron dos capas más, la de dominio, que contiene todos los objetos que hacen referencia a entidades de la realidad del problema y una capa de datos puros, está última contiene básicamente Data Types [4].



Capa UI – User Interfase

Capa de presentación, la lógica de esta capa es totalmente de presentación, se encarga de exponer al usuario los componentes y textos para que pueda interactuar con el sistema.

Capa BL – Business Logic

Capa de negocio, encargada de gestionar las solicitudes de la capa de presentación y realizar las solicitudes correspondientes a la capa de persistencia. Esta capa concentra todo el conocimiento del negocio.

Capa PL – Persistence Logic

Capa encargada de persistir los datos en la base de datos, se encarga de gestionar la relación entre los objetos y tablas de la base de datos, manteniendo independencia del motor de base de datos escogido.

Capa Domain

Capa encargada de modelar el dominio de la realidad planteada. Los objetos de dominio poseen comportamiento sobre las clases relacionadas.

Capa Data

Capa utilizada por las demás para transferir datos entre la lógica y la presentación, las clases de esta capa siguen el patrón Data Transfer Object [5], no poseen comportamiento.

Todos los objetos del dominio tienen su representante en esta capa, los objetos de dominio son accedidos por la persistencia y lógica de negocio, mientras que sus representantes en esta capa son accedidos por la lógica de negocio y la capa de presentación. Los objetos de dominio son procesados a nivel de capa lógica y representados en objetos de datos de la capa data. Los objetos de esta capa son todos serializables [6] ya que deben ser trasferidos desde el servidor web al navegador de los usuarios.

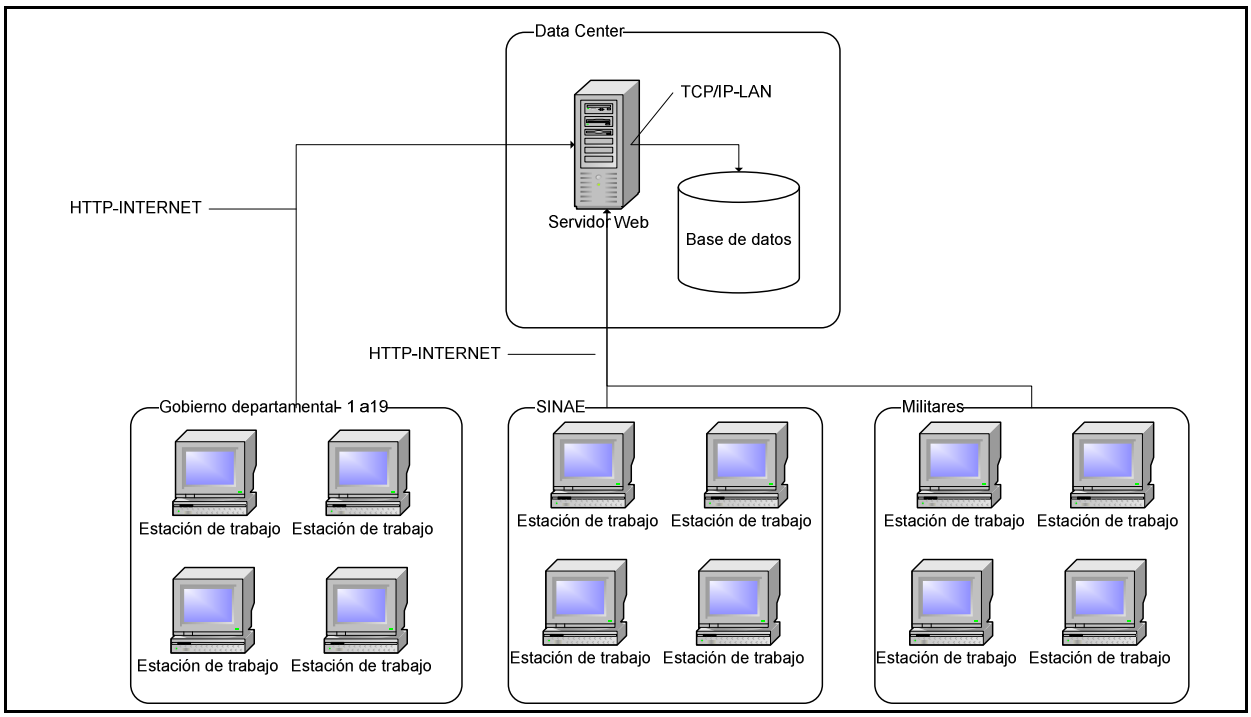
3.1.1. Base de datos

El motor de base de datos escogido es relacional ya que de esa forma utilizamos un Framework llamado Hibernate [7] que permite de forma simple y transparente mantener la estructura de la base de datos, mapeando los objetos del dominio con la estructura necesaria para persistir los objetos. Solo se tiene que tener en cuenta a la hora de configurar dicho Framework que el resultado final del esquema de la base de datos esté en tercera forma normal [8].

3.1.2. Despliegue de la aplicación

Diagrama de despliegue

La siguiente figura exhibe el despliegue de la aplicación en los servidores, mostrando los protocolos y posibles canales de comunicación utilizados entre cada uno de los actores.



Usuarios

Los usuarios del sistema están compuestos por los funcionarios de los 19 gobiernos departamentales que pertenecen a los CECOED y/o colaboran en la atención a un desastre, los funcionarios del SINAE y los militares que apoyan en la operativa.

Comunicaciones

Las comunicaciones con los gobiernos departamentales deben soportarse mediante un enlace estable y rápido a través de la red Internet, pero una mejor solución en cuanto a seguridad y velocidad de acceso sería una red LAN.

Base de datos

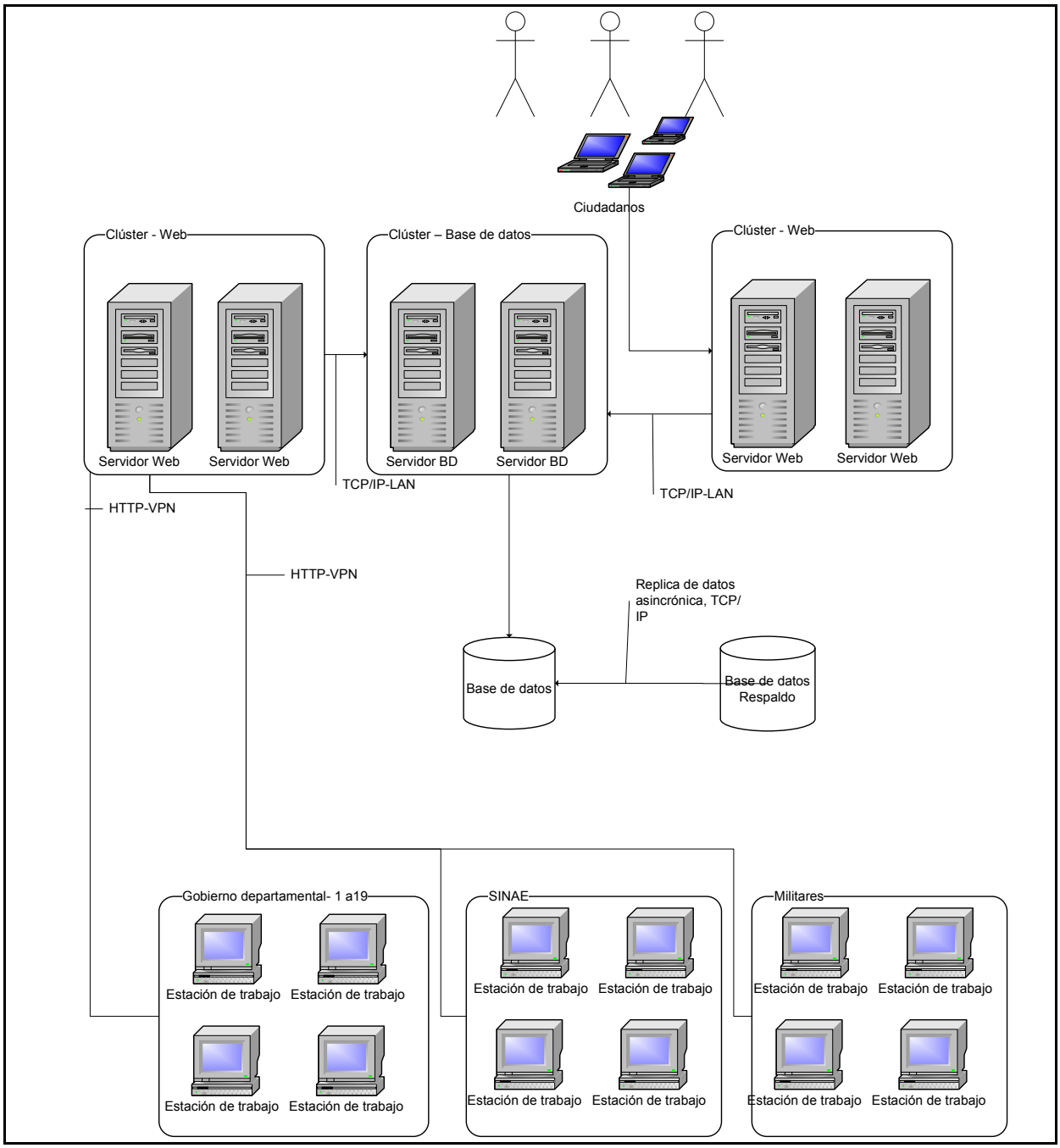
Soporta toda la información relacionada al sistema y especialmente la gestión documental de imágenes, como fotografías de las casas afectadas.

Alta disponibilidad

El sistema de motor de base de datos debe brindar una alta escalabilidad, debido a que el sistema puede crecer mucho, por ejemplo si se comenzara a llevar información del estado de humedad del suelo en distintos puntos del país con reportes cada pocos segundos, la base de datos sería el cuello de botella si no pudiera escalar a una solución más amplia.

En esta misma línea el servidor web también debe poder escalar a una solución que soporte un gran número de usuarios concurrentes dado que por ejemplo en una futura versión, la población podría ingresar al sistema para realizar pedidos de ayuda, denunciar desaparecidos o buscar personas desaparecidas.

A continuación se exhibe un posible diagrama de despliegue de la aplicación en donde se ha escalado a una solución que soporte mayores requerimientos.



Usuarios

Los usuarios están conformados por la población, por los Gobiernos Departamentales a través de los CECOED, SINAE y militares.

Cluster Web

La población es atendida por un cluster de servidores web y los usuarios más críticos que son los CECOED, SINAE y militares por otro cluster web.

De esta forma se tiene tolerancia ante fallos de uno de los servidores web pudiendo continuar online el otro y además se soporta un mayor número de usuarios concurrentes. Esto dependerá de la

memoria RAM con la que cuenten los servidores web, la cantidad de núcleos de los procesadores y sus velocidades, la velocidad de acceso a disco para escritura de los logs de la aplicación y los canales de comunicación.

Cluster Base de Datos

Para mantener una alta disponibilidad, se plantea un cluster a nivel del motor de la base de datos para soportar mayor demanda y a la vez tener tolerancia a fallos de uno de los servidores de base de datos.

Base de Datos de Respaldo

Es conveniente tener una base de datos idéntica a la de producción en todo momento ya que esto permite tener respaldada toda la información y además en caso de que algo le suceda al storage que soporta la base de datos principal, se puede simplemente pasar a trabajar con la de respaldo hasta que se arregle la principal.

4. Modelo de dominio

En esta sección se presenta una descripción de las principales entidades de dominio.

Usuario: Representa a los usuarios del sistema.

Perfil: Conjunto de permisos.

Permiso: Clave de acceso a funcionalidad en el sistema.

PuntoReferencia: Lugares físicos importantes por su función en la gestión de desastres.

Evento: Agrupa toda la información asociada al registro de un evento.

TipoEvento: Los eventos son de distinta naturaleza, esta clase representa los distintos tipos de eventos que pueden ocurrir.

Desastre: Esta clase registra el momento en el que un evento es declarado como desastre y es utilizada para asociar información del desastre en distintos puntos del sistema.

Donación: Agrupa la colección de suministros donados y los datos de donante, fecha y demás.

Suministro: Simboliza un suministro donado o comprado, agrupa la información de vencimiento, refrigeración, cantidad donada o comprada, etc.

TipoSuministro: Representa los distintos tipos de suministros como ser, frazadas, aspirinas, etc.

Depósito: Representación lógica del lugar físico donde se guardan los suministros donados o comprados.

CuentaCorrienteSuministro: Asocia un depósito con un tipo de suministro y lleva la cantidad total disponible.

Parcela: Representa una propiedad siniestrada.

Foto: Entidad que posee la información de una foto de una propiedad siniestrada.

TipoParcela: Las propiedades tienen diferentes usos, está clase simboliza si la priopiedad es un galpón, un depósito, una casa, etc.

DatosVivienda: Entidad que agrupa la información del estado de la vivienda, materiales utilizados en la construcción, si tiene saneamiento, etc.

Unidad: Una parcela cuanta con muchas unidades individuales para las cuales se registra distinta información, agrupa los datos de nivel del agua, metros cuadrados afectados, etc.

ProblemasVivienda: Agrupa toda la información sobre los problemas de la vivienda, como ser grietas en las paredes, humedades, etc.

Hacinamiento: Entidad que asocia a la parcela a la información de cuantas personas familias viven en la casa, cuantos dormitorios tienen para dormir, si la casa se utiliza o no para trabajar, etc.

Inundación: Contiene información del estado de inundación de la propiedad, información histórica sobre otros desastres de inundaciones.

Necesidad: Representa una necesidad que puede ser cubierta localmente o con recursos del SINAE, lleva la información de los tipos de suministros y cantidades necesarias, lugar en donde se necesitan, etc.

GestionNecesidad: Entidad que se utiliza para agrupar información de la gestión realizada para atender una necesidad. Para atender una necesidad se debe decidir si se cubre con recursos de los depósitos o se compran, se debe indicar las cantidades por tipo de suministro.

SolicitudEnvio: Cuando se decide cubrir una necesidad con recursos de los depósitos se generan solicitudes de envío a nombre de los depósitos para que los encargados envíen los suministros a los puntos de entrega, está entidad lleva registro de la información pertinente.

5. Módulos

Las clases del diseño fueron agrupadas en módulos y en diagramas de clases de diseño, de tal forma que pueden ser fácilmente manejables y sea más fácil su visibilidad, también es importante la asignación en módulos coherentes para asignar responsabilidades como controladores de operaciones. Se separaron en los siguientes módulos:

Registro

Este módulo agrupa los requerimientos de registros de todo tipo, en nuestro caso el registro de eventos, en trabajos futuros se podrá extender el módulo con nuevos registros, por ejemplo registrar niveles del agua en ríos.

Desastres

Módulo encargado de los requerimientos sobre desastres, en nuestro caso abarca los requerimientos de consultas del estado del desastre, costos asociados al mismo y la declaración de desastre (proceso por el cual se indica que un evento es considerado como desastre). En futuros trabajos se podrá extender el módulo con nuevos requerimientos como monitoreo de incendios o de inundaciones.

Puntos de referencia

Los puntos de referencia se modelaron en un módulo por separado pensando en trabajos futuros como la realización de un mapa de riesgos donde se resalte en distintas capas los puntos importantes junto con sus ubicaciones en el país, además de manejar características propias del punto de referencia.

Registro de propiedades siniestradas.

Debido a las dimensiones de la información relacionada con el registro de propiedades siniestradas y que contar dicho registro fue un requerimiento explicito aportado por el SINAE, decidimos que esté en un módulo dedicado al requerimiento específico, ya que se manejó como posible que el SINAE lo utilizara en el corto plazo y se deberá ajustar a la visión refinada que éste aporte.

Necesidades

El módulo de necesidades abarca desde el ingreso de necesidades al sistema hasta la gestión de dichas necesidades, se agruparon en este módulo por el fuerte relacionamiento entre sus entidades, en nuestro caso se contempló solo las necesidades de suministros. Puede ser extendido en trabajos futuros con solicitudes de personal capacitado en distintas disciplinas, dinero o cualquier otro tipo de necesidad y extender en consecuencia la gestión que se realiza para cubrir dichas necesidades.

Depósitos

El módulo de depósitos agrupa los requerimientos asociados con la gestión de suministros, esto va desde el ingreso al sistema por los puntos de entrada, la asignación de depósitos, la confirmación del ingreso de suministros al depósito y toda la gestión de solicitudes de envío de suministros a los puntos finales. Puede ser extendido en futuros trabajos agregando por ejemplo mayor control de stock, funcionalidades de ajustes de stock, recepción de compras, entre otros.

Seguridad

Módulo encargado de gestionar la seguridad, tiene como responsabilidad el manejo de usuarios, perfiles y permisos asociados a acciones en el sistema, los demás módulos consultan a éste antes de permitir a un usuario realizar una operación.

Cálculo de indicadores

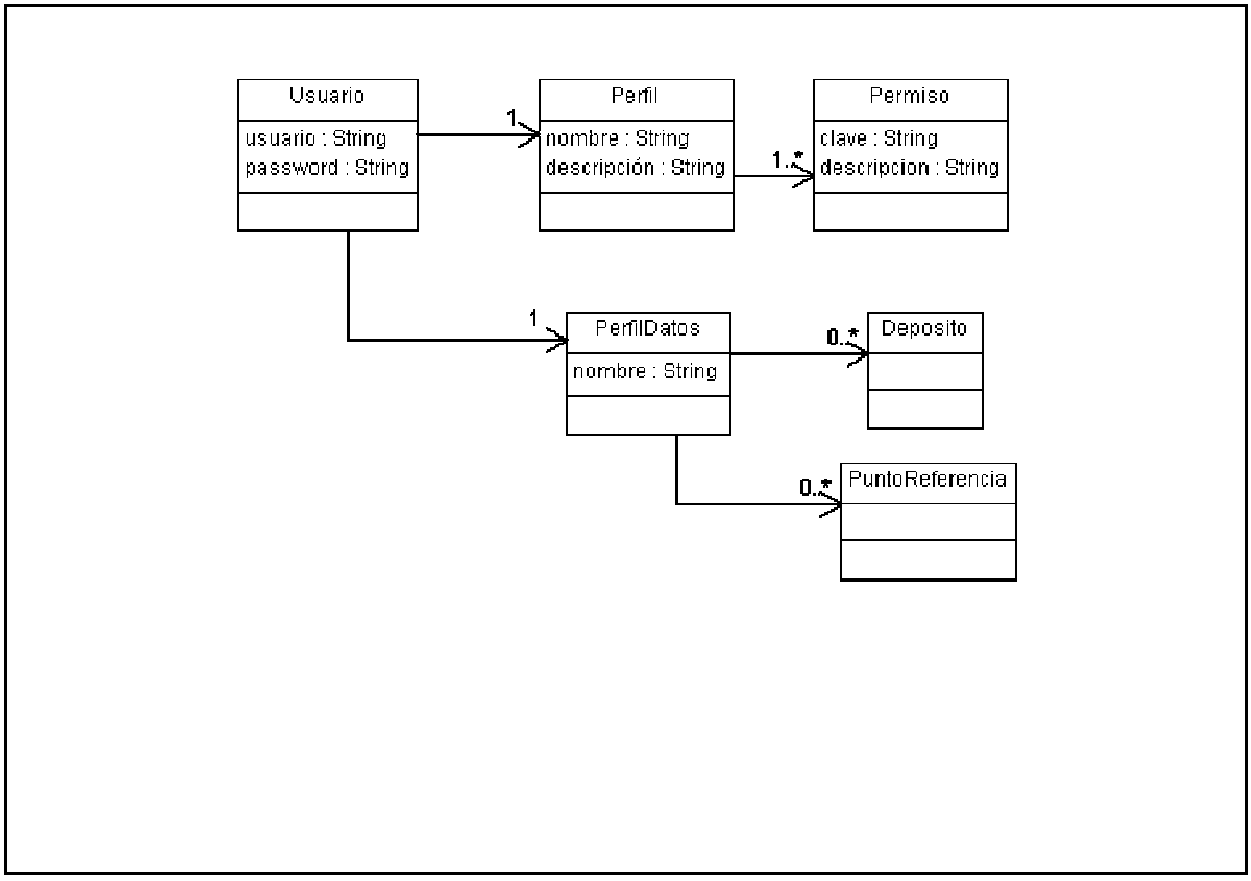
La responsabilidad de este módulo es agrupar y gestionar todo lo referente al cálculo de indicadores que sirvan a la toma de decisiones, en nuestro caso solo abarca dos indicadores de los cuatro desarrollados por el BID e IDEAS llamados IDL e IGR. En futuros trabajos se podría extender el módulo con el cálculo de los dos indicadores restantes y algún otro que se considere importante.

Generales

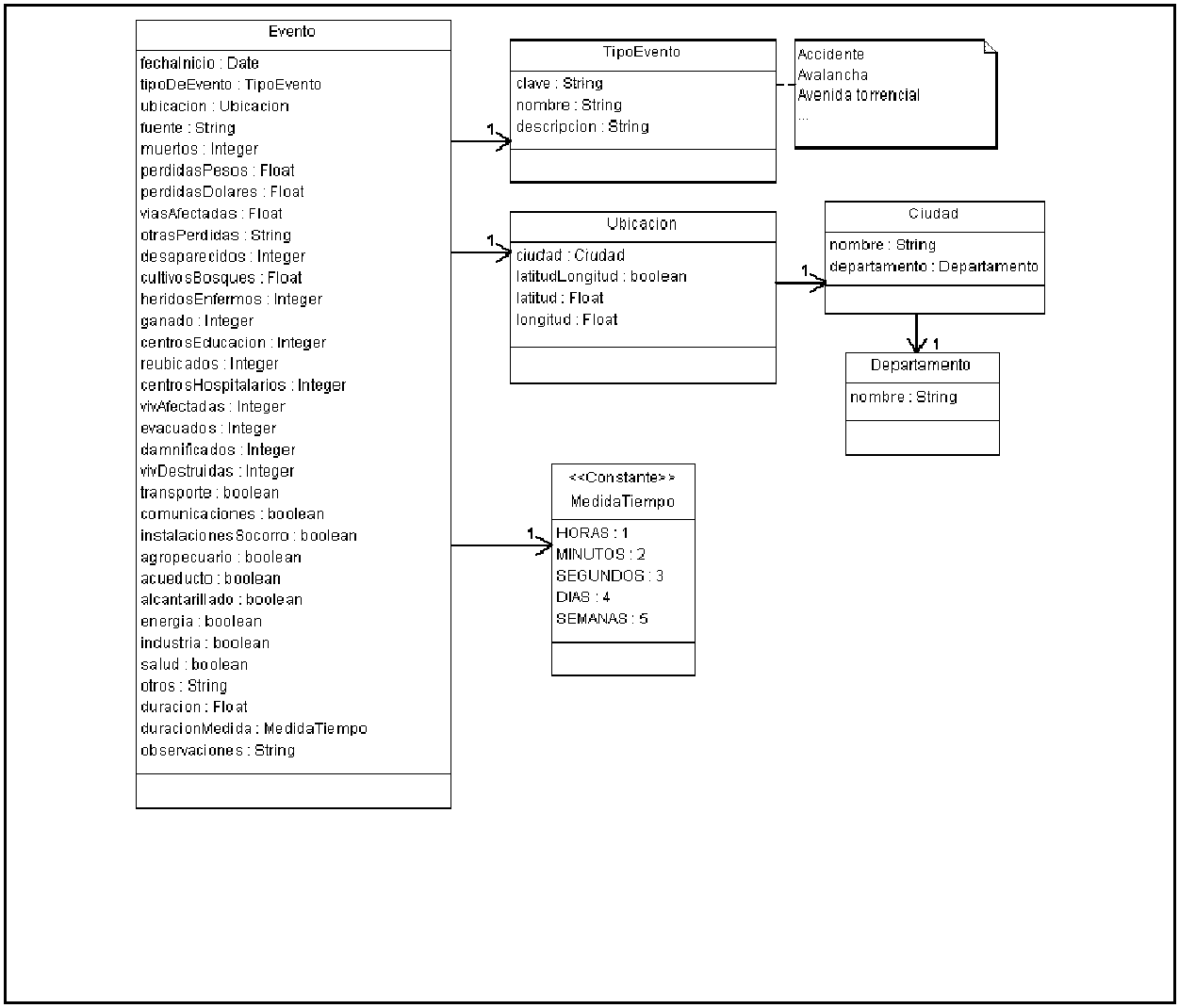
Este módulo agrupa los pequeños conjuntos de datos que por sus dimensiones y posibilidades de extensión no justifican la creación de un módulo dedicado. En nuestro caso asignamos el manejo de ciudades y departamentos a este módulo.

A continuación se exhiben los diagramas UML de clases de diseño para los módulos mas relevantes.

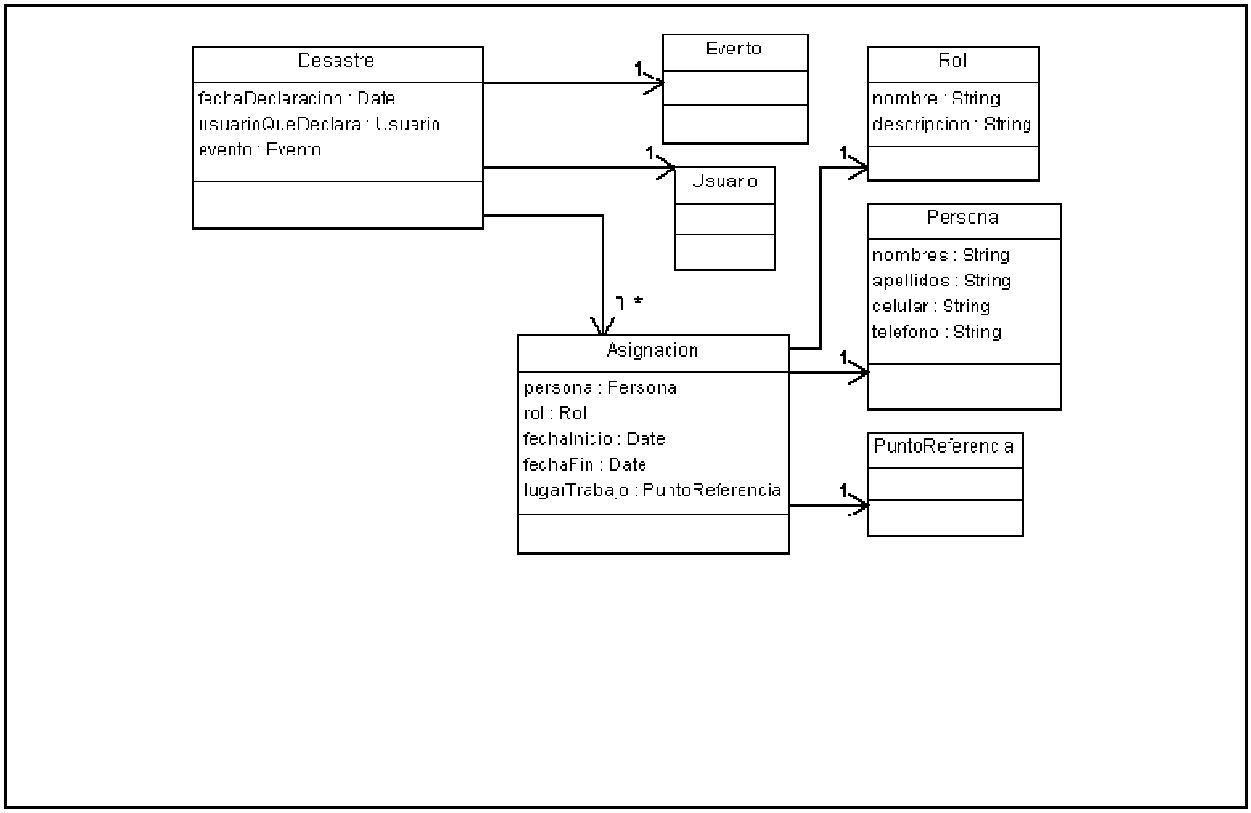
5.1. Clase de diseño modulo usuarios



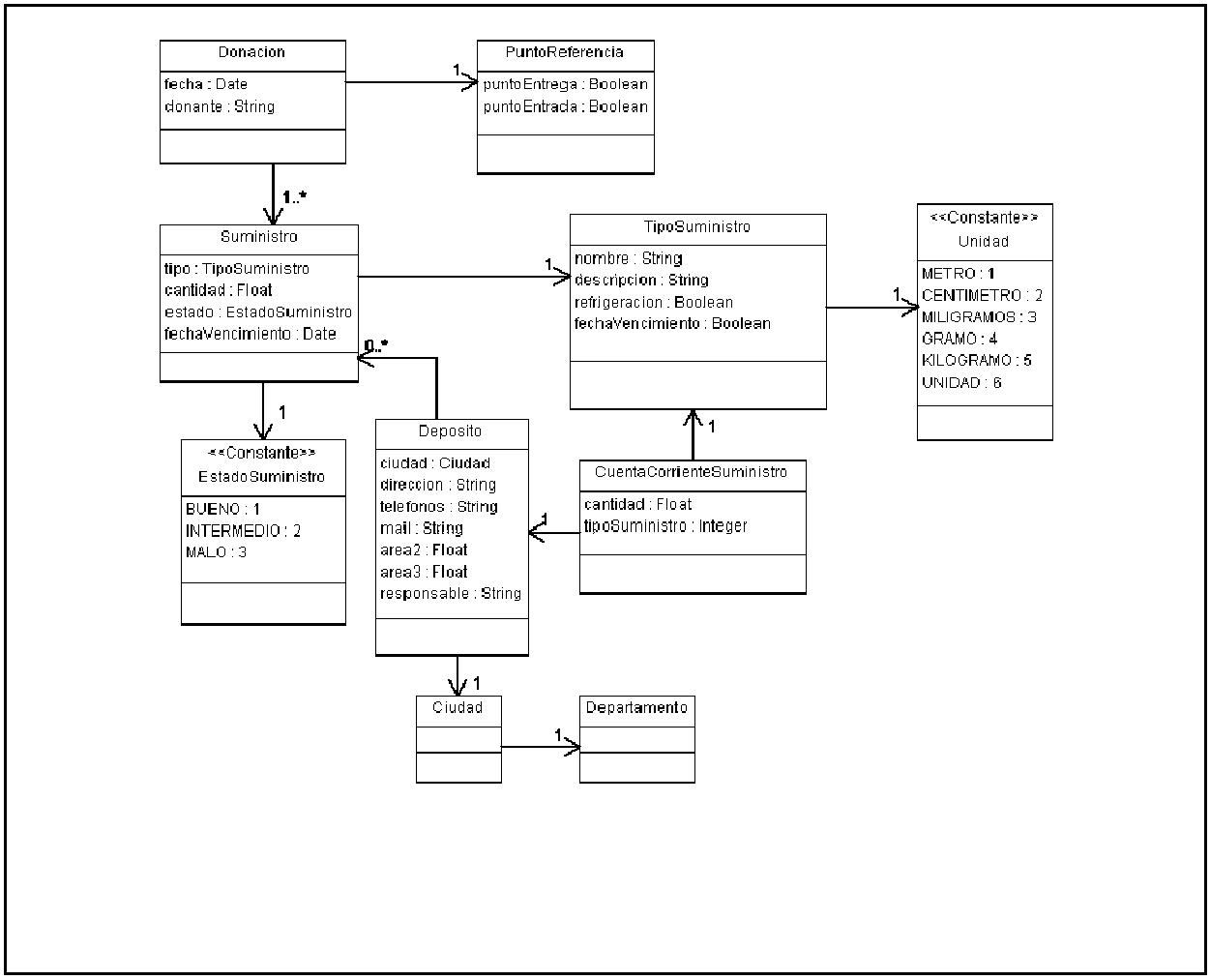
5.2. Clase de diseño modulo registro de eventos



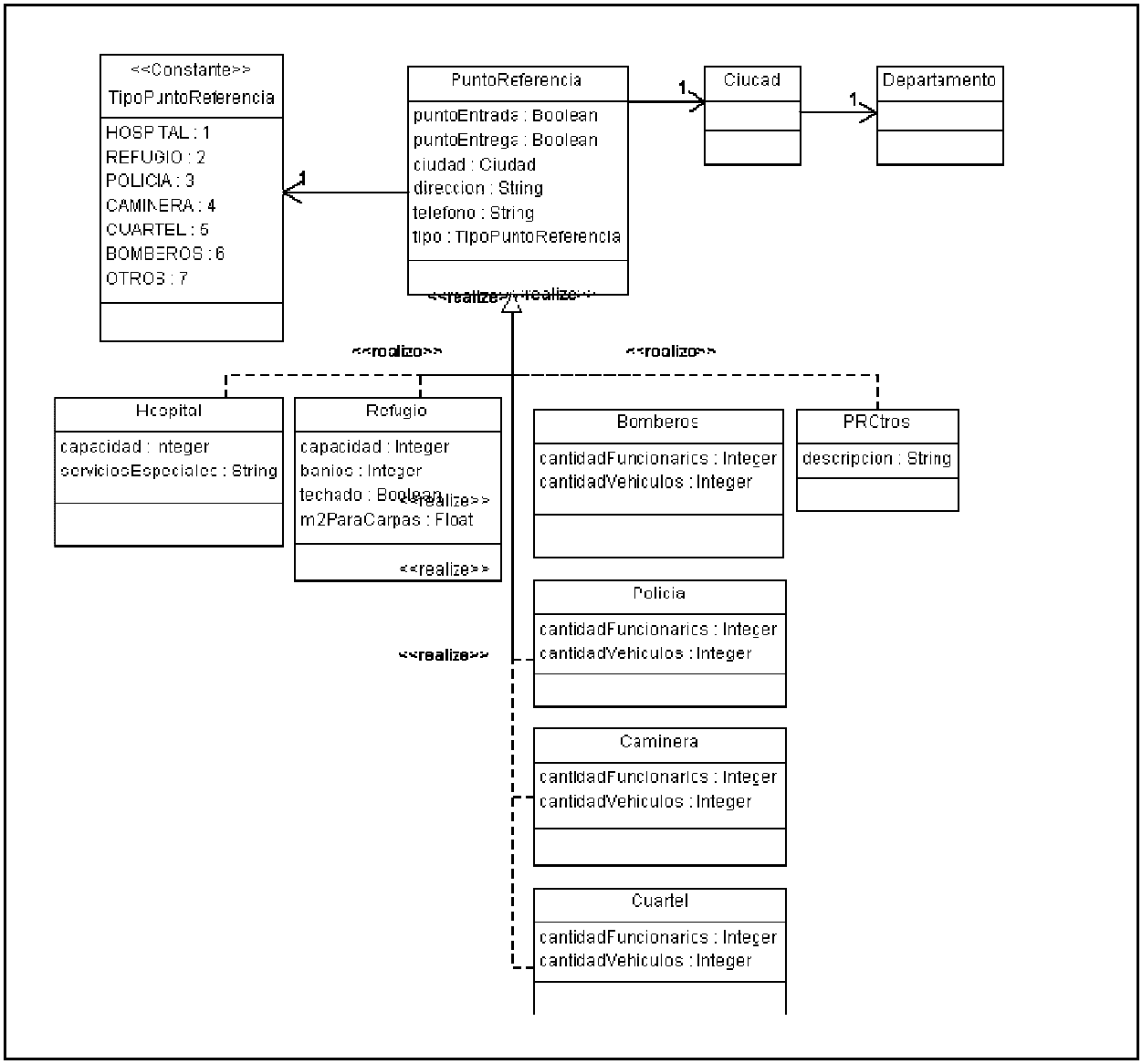
5.3. Clase de diseño modulo desastres



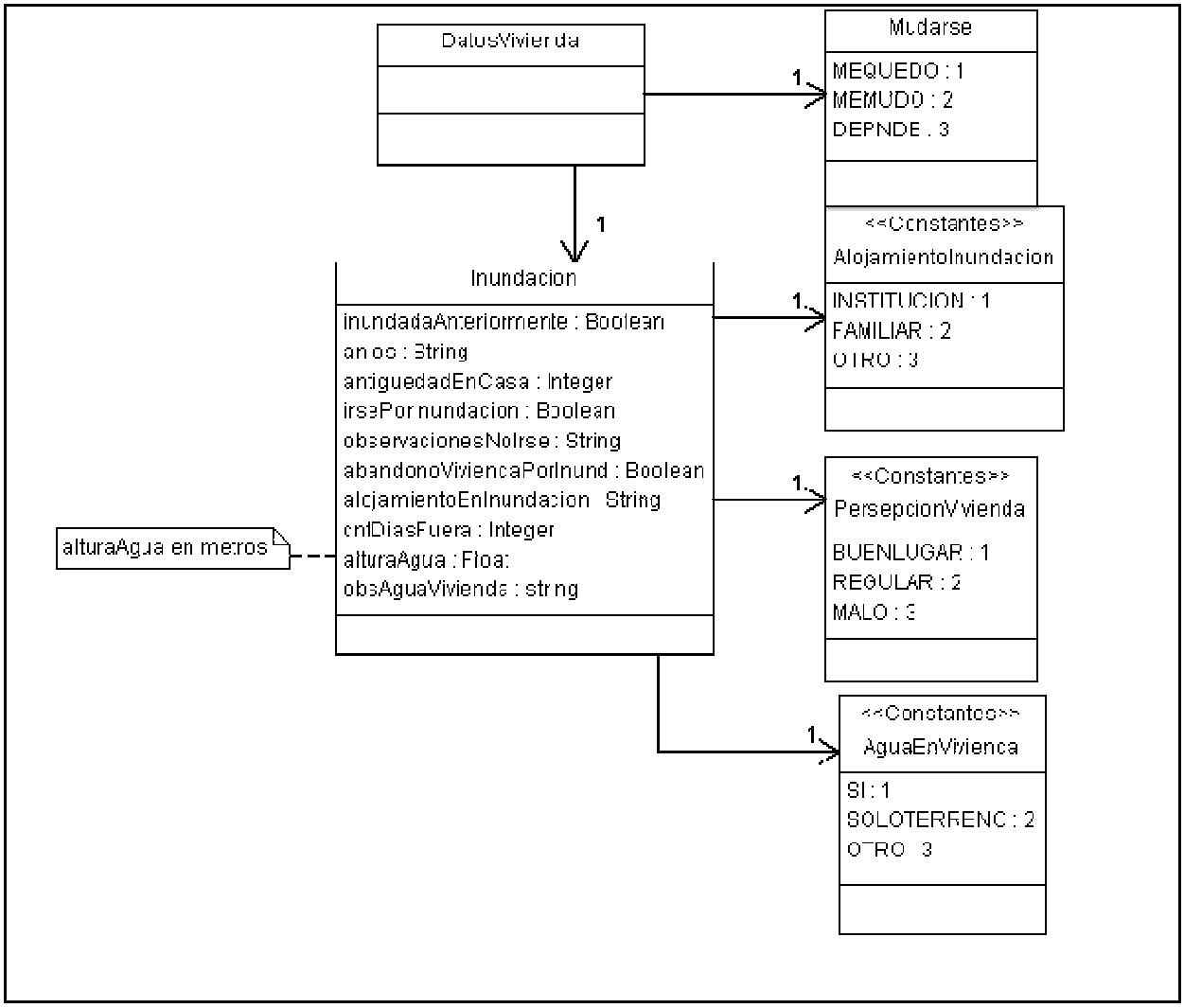
5.4. Clase de diseño modulo suministros

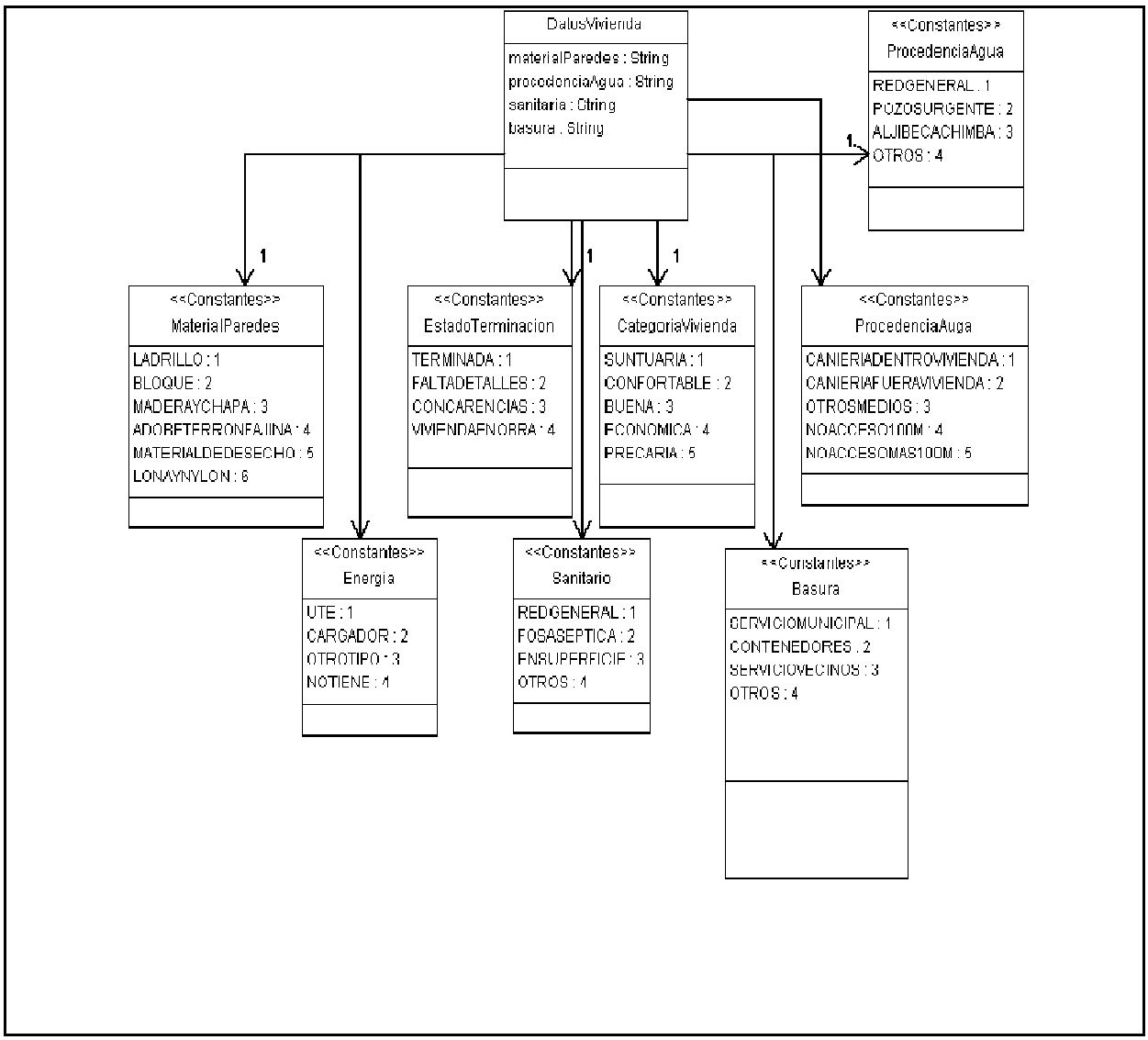


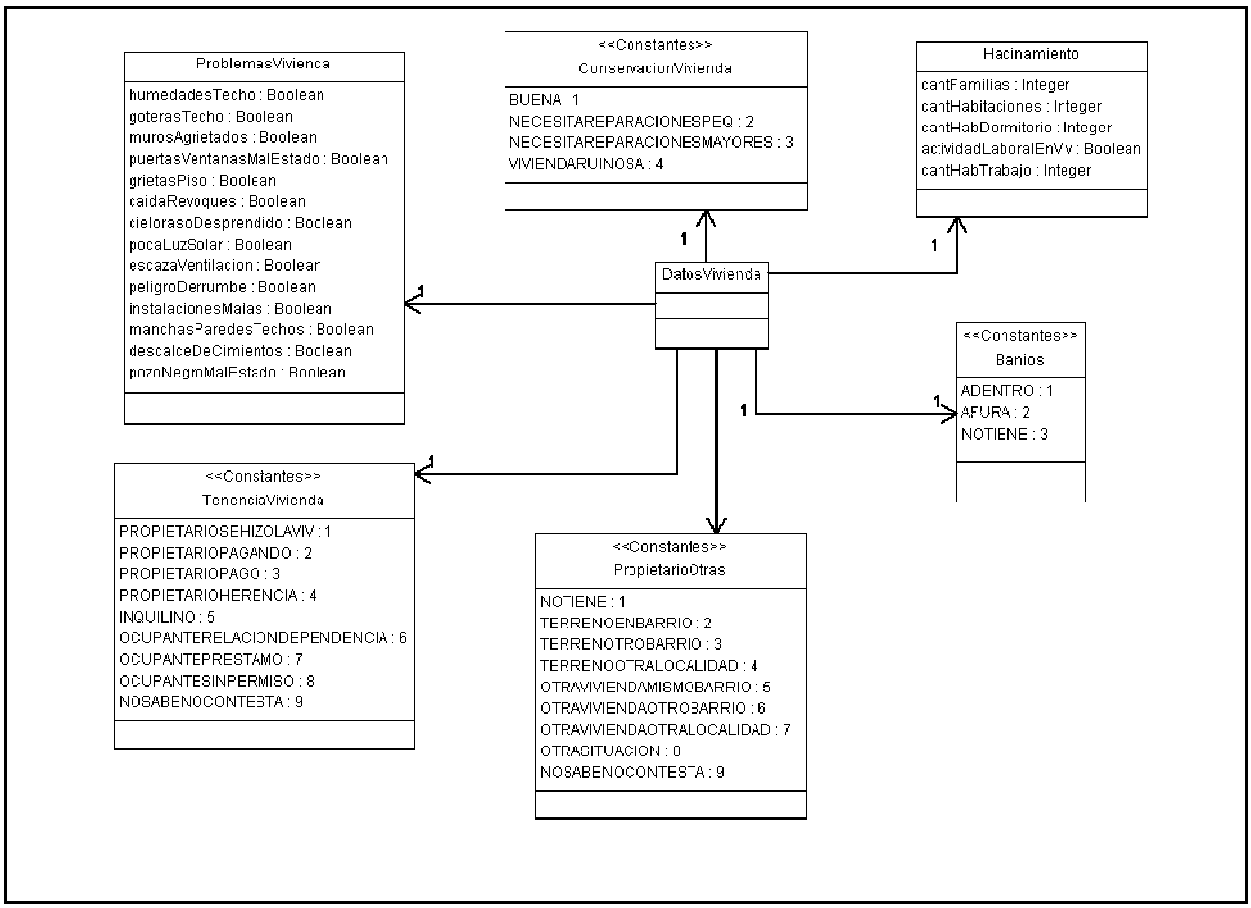
5.5. Clase de diseño modulo punto referencia

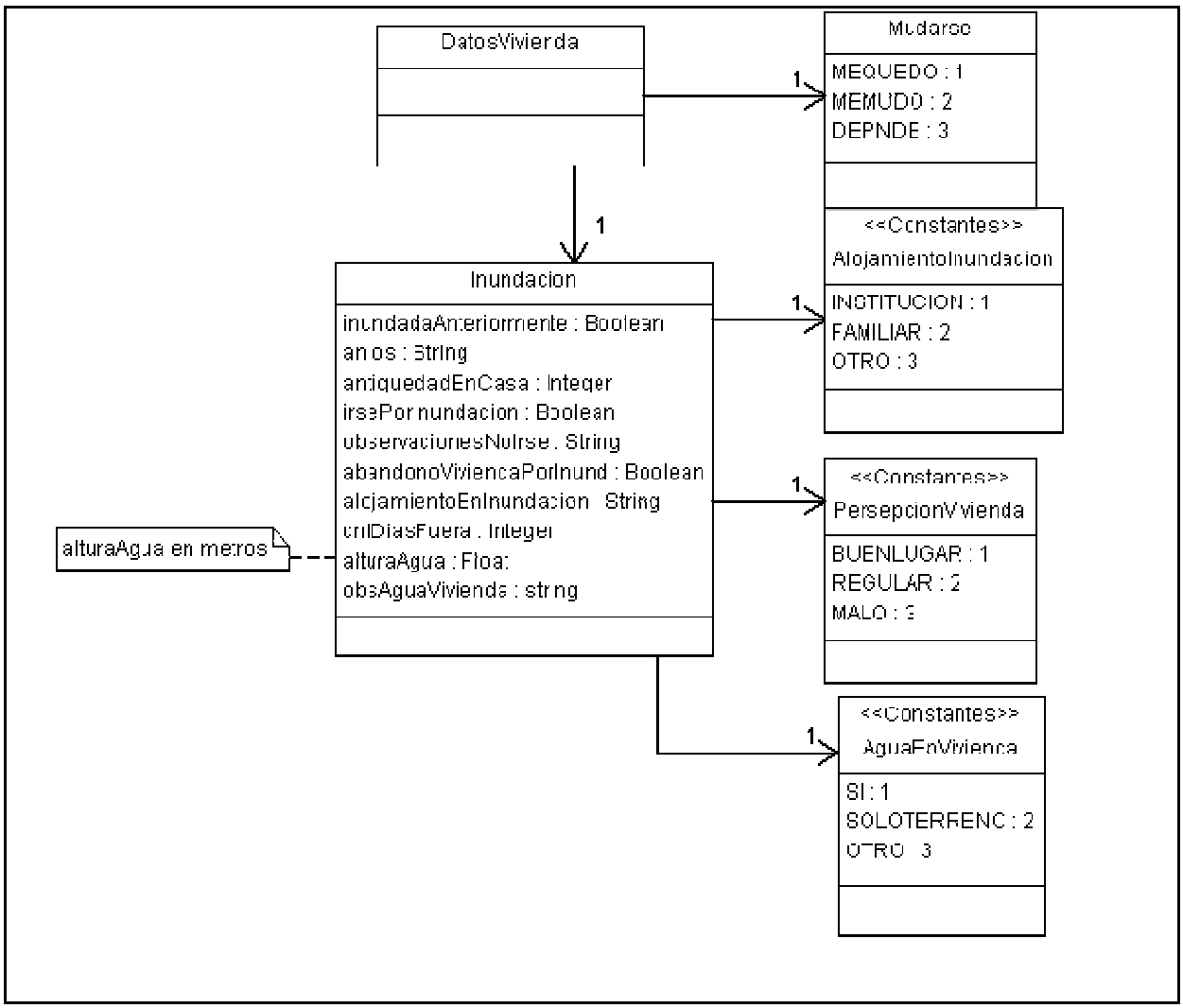


5.6. Clase de diseño modulo registro propiedades siniestradas

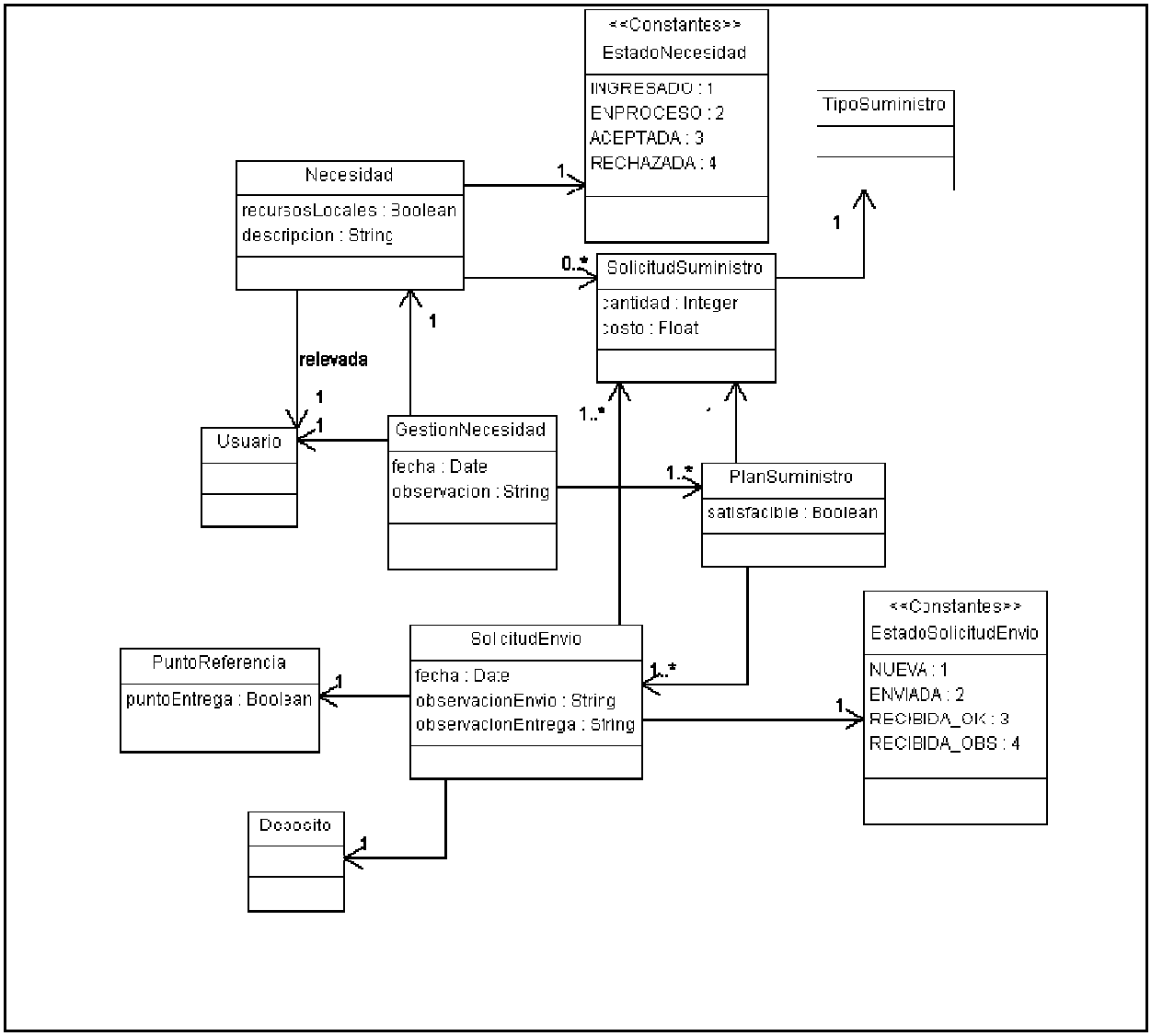



DatosVivienda
materialParedes : String
procedenciaAgua : String
sanitaria : String
basura : String
<<Constantes>>
ProcedenciaAgua
REDGENERAL : 1
POZOSURGENTE : 2
ALJIBECACHIMBA : 3
OTROS : 4
1
1
1
1
<<Constantes>>
MaterialParedes
LADRILLO : 1
BLOQUE : 2
MADERAYCHAPA : 3
ADOBETERRONFALIINA : 4
MATERIALDEDESECHO : 5
LONAYNYLON : 6
<<Constantes>>
EstadoTerminacion
TERMINADA : 1
FALTADETALLES : 2
CONCARENCIAS : 3
VIVIENDAENOBRA : 4
<<Constantes>>
CategoriaVivienda
SUNTUARIA : 1
CONFORTABLE : 2
BUENA : 3
ECONOMICA : 4
PRECARIA : 5
<<Constantes>>
ProcedenciaAuga
CANIERIADENTROVIVIENDA : 1
CANIERIAFUERAVIVIENDA : 2
OTROSMEDIOS : 3
NOACCESO100M : 4
NOACCESOMAS100M : 5
<<Constantes>>
Energia
UTE : 1
CARGADOR : 2
OTROTIPO : 3
NOTIENE : 4
<<Constantes>>
Sanitario
REDGENERAL : 1
FOSASEPTICA : 2
ENSUPERFICIE : 3
OTROS : 4
<<Constantes>>
Basura
SERVICIOMUNICIPAL : 1
CONTENEDORES : 2
SERVICIOVECINOS : 3
OTROS : 4


ProblemasVivienca
humedadesTecho : Boolean
goterasTecho : Boolean
murosAgrietados : Boolean
puertasVentanasMalEstado : Boolean
grietasPiso : Boolean
caidaRevoques : Boolean
cielorasoDesprendido : Boolean
pocaLuzSolar : Boolean
escazaVentilacion : Boolear
peligroDerrumbe : Boolean
instalacionesMalas : Boolean
manchasParedesTechos : Boolean
descalceDeCimientos : Boolean
pozoNegroMalEstado : Boolean
<<Constantes>>
ConservacionVivienda
BUENA 1
NECESITAREPARACIONESPEQ : 2
NECESITAREPARACIONESMAYORES : 3
VIVIENDARUINOSA : 4
Hacinamiento
cantFamilias : Integer
cantHabitaciones : Integer
cantHabDormitorio : Integer
actividadLaboralEnVv : Boolean
cantHabTrabajo : Integer
DatosVivienda
1
1
1
1
<<Constantes>>
Banios
ADENTRO : 1
AFURA : 2
NOTIENE : 3
<<Constantes>>
TenenciaVivienda
PROPIETARIOSEHIZOLAVIV : 1
PROPIETARIOPAGANDO : 2
PROPIETARIOPAGO : 3
PROPIETARIOHERENCIA : 4
INQUILINO : 5
OCUPANTERELACIONDEPENDENCIA : 6
OCUPANTEPRESTAMO : 7
OCUPANTESINPERMISO : 8
NOSABENOCONTESTA : 9
<<Constantes>>
PropietarioOtras
NOTIENE : 1
TERRENOENBARRIO : 2
TERRENOTROBARRIO : 3
TERRENOOTRALOCALIDAD : 4
OTRAVIVIENDAMISMOBARRIO : 5
OTRAVIVIENDAOTROBARRIO : 6
OTRAVIVIENDAOTRALOCALIDAD : 7
OTRASITUACION : 0
NOSABENOCONTESTA : 9


DatosVivienda
Mudarse
MEQUEDO : 1
MEMUDO : 2
DEPNDE : 3
1
1
<<Constantes>>
AlojamientoInundacion
INSTITUCION : 1
FAMILIAR : 2
OTRO : 3
Inundacion
inundadaAnteriormente : Boolean
anos : String
antiquedadEnCasa : Integer
irsePorInundacion : Boolean
observacionesNoIrse : String
abandonoViviencaPorInund : Boolean
alojamientoEnInundacion : String
cntDiasFuera : Integer
alturaAgua : Float
obsAguaVivienda : string
1
<<Constantes>>
PersepcionVivienda
BUENLUGAR : 1
REGULAR : 2
MALO : 3
alturaAgua en metros
1
<<Constantes>>
AguaEnVivienca
SI : 1
SOLOTERRENO : 2
OTRO : 3

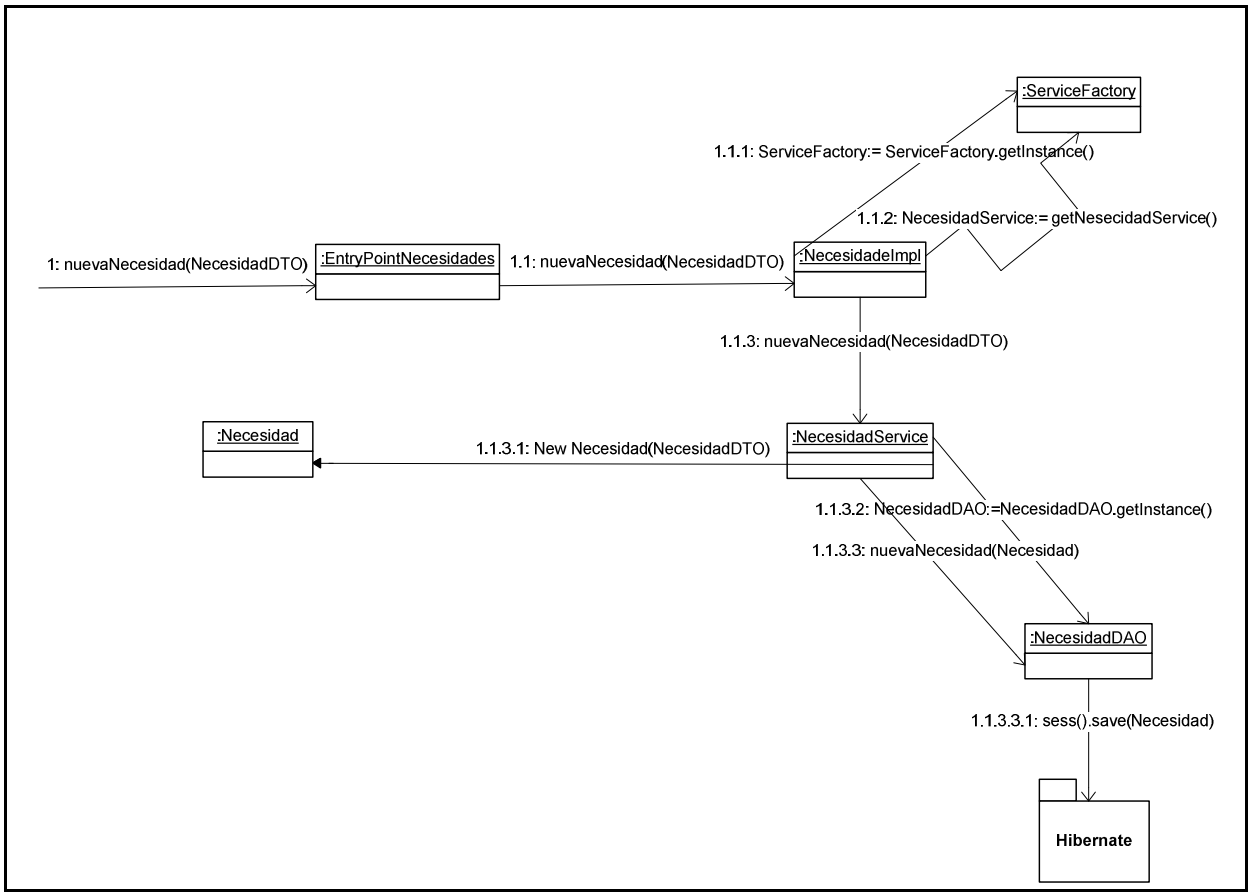
5.7. Clase de diseño modulo necesidades



Como vimos en los requerimientos, dentro de los principales casos de uso se encuentra el alta de necesidades, dicho caso de uso es tomado para ejemplificar todas las colaboraciones del sistema. Se deben seguir los estándares y patrones de diseño, descriptos en el apartado de implementación, como ser Singleton [9], Data Transfer Object [5], Object Data Access [10], Factory [11], entre otros, que ayudan a que las colaboraciones del sistema sean todas similares.

Diagrama de colaboración para el caso de uso alta de necesidad

Para el caso de uso alta de una necesidad es necesario contar con una operación NuevaNecesidad, a continuación se muestra el diagrama de colaboración para esta operación.



El EntryPointNecesidades es un componente perteneciente a la capa de presentación, corre en el navegador del usuario, es quien recibe la solicitud de nueva necesidad. Luego se comunica por HTTP con la clase NecesidadImpl que es la responsable de comunicarse con los servicios de negocio, ésta accede a la fábrica de servicios que sigue el patrón Singleton y luego le solicita que cree la necesidad pasando NecesidadDTO (sigue el patrón Data Transfer Object, clase serializable que pertenece a la capa Data) con la información necesaria para realizar la operación. El servicio de negocio NecesidadService, perteneciente a la capa de negocio, crea la necesidad pasando NecesidadDTO y el constructor de la clase Necesidad es el encargado de traducir la información alojada en NecesidadDTO a sus propios atributos privados. Luego el servicio de negocio se comunica con la capa de persistencia, accede a la clase NecesidadDAO (sigue el patrón Object Data Access y Singleton) y le solicita que persista la necesidad. NecesidadDAO solicita al Framework Hibernate que guarde en la base de datos toda la información asociada a la nueva necesidad.

6. Implementación

6.1. Herramientas

En esta sección se presentan todas las herramientas que se utilizaron para el desarrollo de la aplicación. Dentro de los principales motivos que intervinieron en la decisión sobre que herramientas escoger, está la necesidad de utilizar herramientas de Software Libre, dado que el SINAE solo utiliza estas herramientas. Por otra parte se consideraron herramientas ampliamente aceptadas y utilizadas por la comunidad de desarrolladores y por último se tuvo en cuenta la experiencia con la que cuenta el grupo en cada herramienta.

6.1.1. Lenguaje de programación – Java

Para el desarrollo se decidió utilizar el lenguaje Java ya que para el mismo existen muchas herramientas, frameworks, etc, desarrolladas con licencias de código abierto o de libre uso. Otro punto determinante en la elección del lenguaje de programación fue el conocimiento y experiencia de los dos integrantes del grupo en este lenguaje [12].

6.1.2. Entorno de desarrollo - Eclipse 3.7 Indigo

El entorno de desarrollo (IDE) escogido es el Eclipse 3.7 Indigo, el IDE es de libre uso y está ampliamente aceptado por la comunidad de programadores Java. Principalmente se decidió utilizar este IDE por ser uno de los más conocidos por los integrantes del grupo para el desarrollo con Java, la versión del mismo se eligió tomando en cuenta el pug-in existente para GWT y porque es la última versión del IDE [13].

6.1.3. Desarrollo de interfase de usuarios – GWT

Para el desarrollo de la interfase de usuarios web, se escogió el framework desarrollado por Google, Google Web ToolKit. Este framework permite desarrollar páginas que hacen uso de de la tecnología AJAX en una forma transparente y simple. Básicamente se desarrolla en Java y GWT provee un compilador que traduce las clases Java a código JavaScript que se ejecuta en el navegador del cliente y hace llamadas asincrónicas a los servicios implementados como Servlets de java. Además se cuenta con un plug-in para Eclipse que facilita el desarrollo, depuración y ejecución de las aplicaciones que utilizan GWT. El código logrado con este sistema es muy simple de mantener y entender, hace que el desarrollo de las interfases web sea simple y no hace falta conocer javascript, para incorporar un nuevo desarrollador al equipo, solo tiene como requerimiento que este sepa JAVA y esté familiarizado con Eclipse. Si bien el grupo no conoce GWT considera que es una muy buena herramienta de punta y que es una buena oportunidad para aprender a utilizarla [14].

6.1.4. Servicios de negocio - Spring Business Service

Para el desarrollo de los servicios de negocio se utilizó una herramienta brindada por Spring, ésta permite desarrollar servicios de negocio que siguen el patrón Singleton por lo que son instanciados un única vez. Además está integrado con Hibernate y provee una clase llamada Hibernate Transaction Manager que brinda transnacionalidad (a nivel de base de datos) a los métodos de los servicios de negocio, utilizando Hibernate como framework de persistencia [15].

6.1.5. Persistencia – Hibernate

La persistencia de la aplicación se desarrolló utilizando el framework de persistencia Hibernate 3.0. Este framework permite relacionar de forma muy simple los objetos del dominio con las tablas de la base de datos respetando el diseño de los objetos, permite hacer consultas fáciles de mantener y provee independencia de la base de datos, por lo que es posible cambiando su configuración utilizar otra base de datos.

Provee un lenguaje de consulta llamado HSQL que asegura independencia en la forma en que se forman las consultas para los distintos motores de base de datos. También provee una funcionalidad muy útil para agilizar el desarrollo, que es el mantenimiento automático del esquema de la base de datos [7].

6.1.6. Reportes – Jasper

Jasper es una herramienta que permite desarrollar de forma rápida reportes y presentarlos en distintos formatos de forma fácil. Jasper también cuenta con un diseñador de reportes IReports el cual se utilizará en su versión 4.1.1 que es la última publicada. Se decidió utilizar esta herramienta porque ya cuenta con muchos años de desarrollo por lo que es muy estable y tiene una fuerte integración con java, además los integrantes del grupo ya poseen experiencia en el uso de la misma [16].

6.1.7. Control de versiones – Subversion

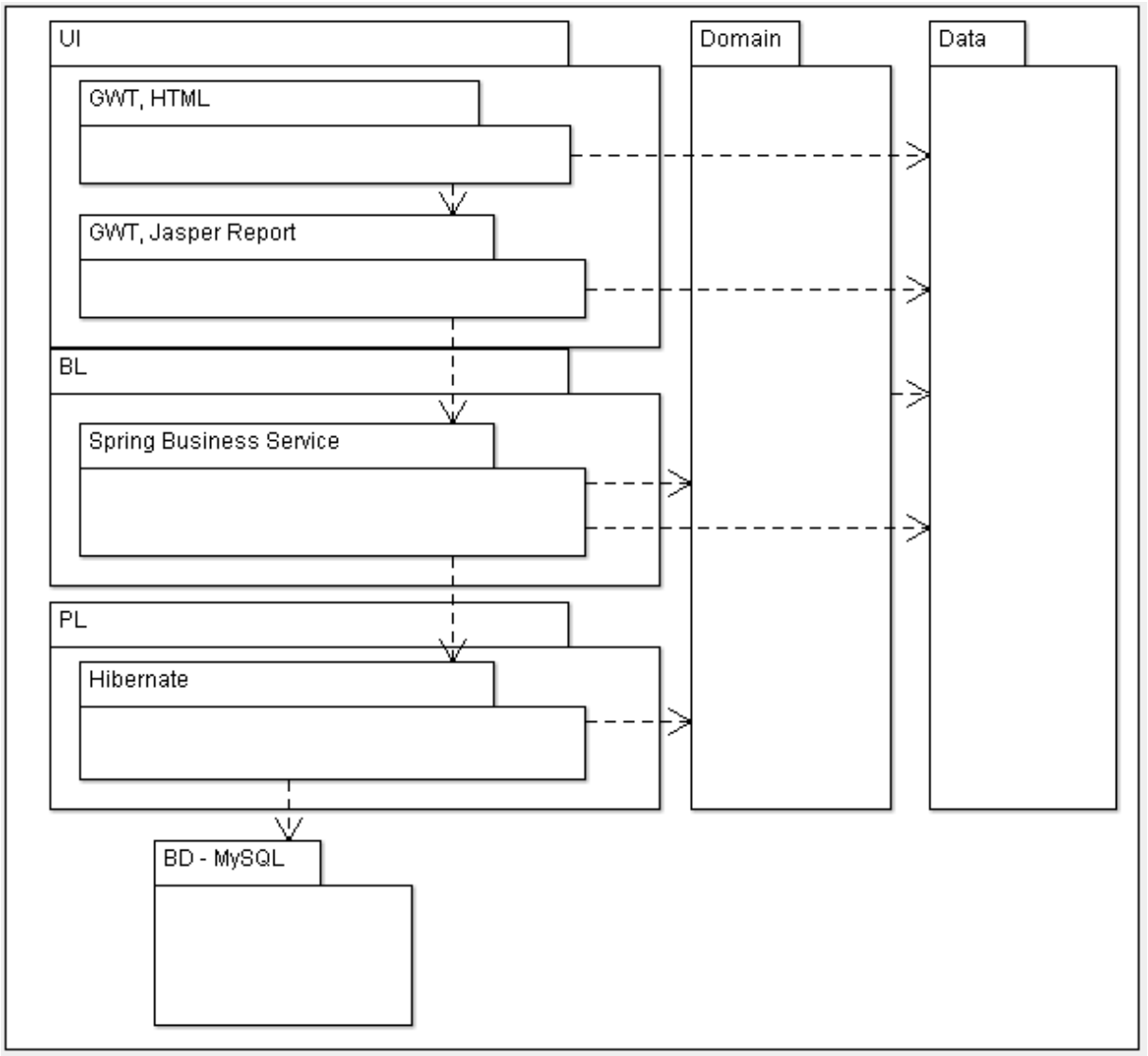
La herramienta elegida para el control de versiones del código fuente y de todos los documentos del proyecto es SVN – Subversion. Se creó un sitio en googlecode http://satode.googlecode.com/svn/trunk en donde se versionan todos los documentos no solo el código fuente. El cliente que se escogió es el TortoiseSVN 1.7.1, es uno de los clientes SVN más difundidos de libre uso y además el grupo ya cuenta con experiencia en su uso [17][18].

6.1.8. Base de datos - MySQL

El motor de base de datos escogido es MySQL Server 5.1.50. Este motor de base de datos es de libre uso, ampliamente aceptado y utilizado por la comunidad de desarrolladores. La versión se escogió porque uno de los integrantes del grupo la utilizó en un proyecto de características similares y respondió muy bien [19].

6.2. Tecnología aplicada a la arquitectura

En la presente sección se exhibe un diagrama de la arquitectura de la aplicación en donde se expone para cada capa las tecnologías de software que serán utilizadas, luego se explica brevemente cada capa.



Capa UI – User Interfase

HTML: Páginas web HTML.

Jasper Report: Reportes Jasper, en formatos pdf, rtf, doc, xlsl y html.

GWT: Se encarga de inicializar los componentes de las páginas, escuchar los eventos de las páginas y hacer las solicitudes a la capa de negocio.

Capa BL – Business Logic

Spring Business Service: Servicios de negocio que siguen el patrón Singleton para asegurar una única instancia del servicio a nivel de toda la aplicación. Controlan las transacciones de base de datos de las operaciones.

Capa PL – Persistence Logic

Hibernate: Framework de persistencia, realiza las actualizaciones, consultas, inserciones, etc., en la base de datos.

6.3. Planificación

Luego del análisis de requerimientos se procedió a realizar el plan de desarrollo del prototipo, donde se definió los puntos a desarrollar cada semana por cada programador, el cual se encuentra como adjunto a este documento como Alcance del Sistema. En dicho plan se tomó en cuenta, que primero

se tienen que desarrollar los módulos que no dependan de otros y luego los que dependen de los anteriores y así hasta completar todos los módulos. En la asignación de horas de programación, se consideró que cada programador se encargará por completo de un módulo, para no tener conflictos en el desarrollo. Como plan de mitigación de atrasos se decidió, que si se encontraba que el desarrollo estaba atrasado una semana o más, la siguiente semana se dedicaban más horas que las planificadas para ponerse al día.

6.4. Patrones y estándares

A continuación se definen los estándares y patrones a seguir. Para los nombres en todos los casos se utilizará el estándar de nomenclatura Camel Case [20].

A continuación se listan en detalle cada uno de los estándares y patrones.

Data Access Object (DAO)

Los Data Access Object son los objetos encargados de la persistencia de una entidad del dominio, cada uno se encarga de implementar los métodos necesarios para persistir o recuperar una determinada entidad del dominio. Por cada entidad de dominio persistible existe un DAO y estos siguen el patrón Singleton. La nomenclatura utilizada para nombrarlos sigue el patrón: nombre de entidad + DAO (Ejemplo: CiudadDAO)

Para los métodos más comunes como nuevo, modificar, eliminar, buscar todos y buscar según un filtro se siguen los siguientes patrones:

- nuevo + nombre de la entidad, método encargado de persistir el nuevo objeto en la base de datos.
- modificar + nombre de la entidad, método encargado de modificar el objeto en la base de datos.
- eliminar + nombre de la entidad, método encargado de eliminar el objeto de la base de datos.
- lista + nombre de la entidad, método encargado de extraer de la base de datos la totalidad de registro de la entidad.
- buscar + nombre de entidad, método encargado de la extracción de los registros asociados a una entidad, que cumplan una determinada condición de filtro.

Servicios de negocio

Los servicios de negocio son los encargados de proveer funcionalidad a la presentación de un determinado módulo, se utilizan para bajar el acoplamiento del sistema. Estos también son encargados de pasar los objetos de dominio a Data Transfer Object (DTO). Todos los servicios de negocio son accedidos a través de una Factory y siguen el patrón Singleton. Solo los servicios de presentación de su módulo acceden a ellos y también prestan funcionalidad a los demás servicios de negocio. De esta forma se reutiliza el código y el conocimiento, se hace más simple el mantenimiento y la legibilidad de los mismos. La nomenclatura utilizada para nombrarlos sigue el siguiente patrón: nombre de entidad principal + Service (Ejemplo: DepositoService).

Para los métodos más comunes como nuevo, modificar, eliminar, buscar todos y buscar según un filtro se siguen los siguientes patrones:

- nuevo + nombre de la entidad, método encargado de solicitar al correspondiente DAO persistir el nuevo objeto en la base de datos.
- modificar + nombre de la entidad, método encargado de solicitar al correspondiente DAO modificar el objeto en la base de datos.

- eliminar + nombre de la entidad, método encargado de solicitar al correspondiente DAO eliminar el objeto de la base de datos.

- lista + nombre de la entidad, método encargado de solicitar al correspondiente DAO extraer de la base de datos la totalidad de registro de la entidad.

- buscar + nombre de entidad, método encargado de solicitar al correspondiente DAO la extracción de los registros asociados a una entidad que cumplan una determinada condición de filtro.

Servicios de presentación

Los servicios de presentación son los encargados de proveer acceso a los servicios de negocio. Estos solo son accedidos desde los Rich Client Application [21]. Pertenecen a un módulo y solo utilizan los servicios de negocio de dicho módulo. La nomenclatura utilizada para nombrarlos sigue el siguiente patrón: nombre de entidad principal + Impl (Ejemplo: DepositoImpl). Para los métodos más comunes como nuevo, modificar, eliminar, buscar todos y buscar según un filtro se siguen los siguientes patrones:

- nuevo + nombre de la entidad, método encargado de solicitar al correspondiente servicio de negocio persistir el nuevo objeto en la base de datos.

- modificar + nombre de la entidad, método encargado de solicitar al correspondiente servicio de negocio modificar el objeto en la base de datos.

- eliminar + nombre de la entidad, método encargado de solicitar al correspondiente servicio de negocio eliminar el objeto de la base de datos.

- lista + nombre de la entidad, método encargado de solicitar al correspondiente servicio de negocio extraer de la base de datos la totalidad de registro de la entidad.

- buscar + nombre de entidad, método encargado de solicitar al correspondiente servicio de negocio la extracción de los registros asociados a una entidad que cumplan una determinada condición de filtro.

Rich Client Application

Clases encargadas de gestionar los componentes de presentación, estas son traducidas a Javascript [22] y se ejecutan en el navegador del usuario, se comunican con los servicios de presentación mediante llamadas asincrónicas, pueden acceder a los servicios de presentación de distintos módulos. La nomenclatura utilizada sigue el siguiente patrón: EntryPoint + nombre de entidad principal (Ejemplo: EntryPointDeposito), la entidad principal es la que gestiona el Rich Client Application.

Paginas HTML

Son encargadas de la maquetación de la presentación, éstas utilizan los Rich Client Application, solo utilizan uno y éste pertenece a su mismo módulo. La nomenclatura utilizada sigue el siguiente patrón: nombre de entidad principal (Ejemplo: Deposito), la entidad principal es la que gestiona el Rich Client Application que utiliza dicha página HTML.

Data Transfer Object

Estos objetos son los representantes de las entidades de dominio, con la salvedad de que ellos y toda la información que encapsulan es serializable, esto quiere decir que puede viajar desde la lógica de negocio hasta los Rich Client Application. El estándar seguido para estos objetos es el siguiente:

- Implementan la interfase java.io.Serializable.
- Implementan la interfase com.google.gwt.user.client.rpc.IsSerializable.
- Todos sus atributos se definen al inicio.
- Solo posee el constructor por defecto.
- Posee un método “set + nombre atributo” y otro “get+ nombre atributo”, por cada atributo.

Dominio

Las entidades de dominio representan objetos de la realidad y son persistibles en la base de datos, éstas siguen el siguiente estándar:

- Todos sus atributos se definen al inicio, todas las entidades de dominio poseen el atributo ID de tipo java.lang.Long.
- El siguiente método es el constructor por defecto que no posee sentencias en su sección de código.
- Luego el constructor que recibe como parámetro el correspondiente DTO.
- Seguido de un método “set + nombre atributo” y otro “get + nombre atributo”, por cada atributo.
- Por último el método getDTO que no recibe parámetros y retorna el correspondiente DTO, esta regla se sigue para que el agregar o quitar atributos a una entidad de dominio solo impacte en este método y en el constructor que recibe el DTO.

6.5. Procedimientos

Desarrollo

Cuando un programador comienza una nueva sesión de desarrollo, sigue el siguiente procedimiento:

- Se actualizan todos los fuentes desde el servidor SVN.
- Se desarrolla.
- Se versionan todos los cambios.

No se termina la sesión de desarrollo hasta completar todos los pasos, esto asegura una buena política de respaldo de los fuentes ya que queda la copia local más la del servidor.

Versionado de los cambios

Al finalizar una sesión de desarrollo los programadores tienen que versionar los cambios, aquí se sigue la siguiente política.

- Todos los fuentes versionados son compilables y se versionan la totalidad de los cambios.

7. Referencias

[1] Crai Larman; *UML y PATRONES. Introducción al análisis y diseño orientado a objetos;* PRENTICE HALL;1999;pp 185 al 215 (ISBN 970-17-0261-1)

[2] Servicios Informáticos. Universidad de Valencia; *Red privada virtual*; http://www.uv.es/siuv/cas/zxarxa/vpn.htm; Último acceso: 27/04/2012

[3] Internetlab; *¿Qué significa el titulo https y cómo funciona?*; http://www.internetlab.es/post/888/que-significa-el-protocolo-https-y-como-funciona/; Último acceso: 27/04/2012

[4] Programación 4, UdelaR*; Programación 4. Conceptos básicos de orientación a objetos(parte 1)*; http://www.fing.edu.uy/inco/cursos/prog4/field.php/Material/Teorico?action=download&upname=03-conceptos_basicos_1era_parte.pdf; Último acceso : 27/04/2012

[5] Core J2EE Patterns – Transfer Object; http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html; Último acceso: 08/04/2012

[6] *Capítulo II – Serialización: Persistencia de datos en Java*; http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/montero_g_g/capitulo2.pdf; Último acceso: 27/04/2012

[7] Hibernate 3.0; http://www.hibernate.org/; Último Acceso: 30/09/11

[8] Fundamentos de base de datos, UdelaR; *Fundamentos de base de datos Tema 4 - Diseño relacional*; http://www.fing.edu.uy/inco/cursos/bdatos/teorico/7-Normalizacion.pdf; Último acceso: 27/04/2012

[9] Microsoft; *El patrón Singleton;* http://msdn.microsoft.com/es-es/library/bb972272.aspx#EDAA; Último acceso: 27/04/2012

[10] Data Access Object; http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html; Último Acceso: 08/04/2012

[11] Microsoft; *Exploring the Factory Design Pattern*; http://msdn.microsoft.com/en-us/library/ee817667.aspx; Último acceso: 27/04/2012

[12] JAVA; http://www.oracle.com/technetwork/java/index.html; Último acceso: 24/09/11

[13] Eclipse 3.7 Indigo; http://eclipse.org/indigo/; Último acceso: 27/09/11

[14] GWT – Google Web Toolkit; http://code.google.com/intl/es-ES/webtoolkit/; Último acceso: 30/09/11

[15] Spring Services – Hibernate; http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/orm.html; Último acceso: 30/09/11

[16] JasperSoft; http://jasperforge.org/projects/jasperreports; Último Acceso: 03/10/11

[17] Subversion; http://es.wikipedia.org/wiki/Subversion; Último Acceso: 03/10/11

[18] Tortoise SVN; http://tortoisesvn.tigris.org/; Último Acceso: 05/10/11

[19] MySQL; http://www.mysql.com/; Último Acceso: 18/11/11

[20] Camel Case; http://msdn.microsoft.com/en-us/library/x2dbyw72%28v=vs.71%29.aspx; Último acceso: 08/04/2012

[21] Rich Client; http://es.wikipedia.org/wiki/Cliente_pesado; Último acceso: 08/04/2012

[22] JavaScript; http://www.w3schools.com/js/js_intro.asp; Último acceso: 08/04/2012

8. Anexos

8.1. Alcance del Sistema

8.1.1. Introducción

En este documento se presenta la definición del sistema a construir.

8.1.2. Propósito

Este documento describe el alcance del sistema en términos de los requerimientos que se implementarán.

Así mismo ayudará en el proceso de organización del proyecto, en función de los tiempos y recursos de los que se dispone.

El alcance definido servirá como entrada de las siguientes actividades:

- Planificación del Proyecto
- Ajustar y Controlar el Desarrollo

8.1.3. Alcance

El alcance del sistema influirá en gran parte del proceso y particularmente en las fases de diseño y construcción.

No pretende ser una extensión detallada de los requerimientos, sino una reseña del subconjunto de éstos que se consideran dentro del alcance del prototipo, dadas las restricciones de recursos y prioridades expresadas por los tutores.

Por detalles acerca de los requerimientos, ver el documento ESPECIFICACIÓN DE REQUERIMIENTOS.doc.

8.1.4. Planificación para lograr el alcance

Fase Diseño

Iteración 1 – Semanas del 04/10/2011 al 30/10/2011

- Realizar documento con la Arquitectura del Sistema.
- Selección de herramientas a utilizar.
- Realizar pruebas de concepto exitosas de todas las tecnologías novedosas a utilizar.
- Realizar un prototipo de la Arquitectura del sistema

Iteración 2 – Semanas del 30/10/2011 al 25/11/2011

- Manejo de Usuarios: Se diseñarán altas, bajas y modificaciones de usuarios y perfiles de usuarios.
- Registro de Eventos: Se diseñarán altas, bajas y modificaciones de eventos.
- Indicación de Desastre: Se podrá indicar que un evento se convirtió en desastre, para luego realizar consultas sobre el desastre, se diseñará su implementación.
- Gestión de suministros

    - Puntos de entrada: Se diseñarán altas, bajas y modificaciones de puntos de entrada de suministros.
    - Puntos finales: Se diseñarán altas, bajas y modificaciones de puntos de finales, donde se entregaran los suministros a los necesitados.
    - Suministros: Se diseñarán altas, bajas, modificaciones de tipos de suministros.
    - Clasificación de suministros: Se diseñarán altas, bajas, modificaciones de suministros
    - Depósitos: Se diseñarán altas, bajas y modificaciones de depósitos, recepción de compras, recepción de donaciones.
- Puntos de Referencia: Se diseñarán altas bajas y modificaciones de puntos de referencia.
- Ingreso de necesidades: Se diseñarán altas bajas y modificaciones de necesidades de suministros.
- Gestión de necesidades: Se diseñaran altas, baja, modificación de solicitudes de envío, marcar como satisfacible o no una necesidad, dar prioridad a una necesidad.
- Registro de propiedades siniestradas: Se diseñaran altas, baja, modificación de registros de propiedades siniestradas.
- Costos: Se diseñaran altas, baja, modificación de costos y tipos de costos.
- Calculo de índices: Se diseñará el cálculo del índice IDL e IGR.

Fase Construcción

Iteración 1 – Semanas del 26/11/2011 al 10/12/2011

- Manejo de Usuarios: Se implementarán altas, bajas y modificaciones de usuarios y perfiles de usuarios.
- Registro de Eventos: Se implementarán altas, bajas y modificaciones de eventos.

Iteración 2 – Semanas del 10/12/2011 al 24/12/2011

- Indicación de Desastre: Se implementará la declaración de desastre.
- Gestión de suministros
    - Puntos de entrada: Se implementarán altas, bajas y modificaciones de puntos de entrada de suministros.
    - Puntos finales: Se implementarán altas, bajas y modificaciones de puntos de finales, donde se entregaran los suministros a los necesitados.
    - Suministros: Se implementarán altas, bajas, modificaciones de tipos de suministros.
    - Clasificación de suministros: Se implementarán altas, bajas, modificaciones de suministros
- Depósitos: Se implementarán altas, bajas y modificaciones de depósitos, recepción de compras, recepción de donaciones.

Iteración 3 – Semanas del 26/12/2011 al 06/01/2012

- Puntos de Referencia: Se implementarán altas bajas y modificaciones de puntos de referencia.
- Ingreso de necesidades: Se implementarán altas bajas y modificaciones de necesidades de suministros.

Iteración 4 – Semanas del 23/01/2012 al 05/02/2012

- Gestión de necesidades: Se implementarán altas, baja, modificación de solicitudes de envío, marcar como satisfacible o no una necesidad, dar prioridad a una necesidad.
- Registro de propiedades siniestradas: Se implementarán altas, baja, modificación de registros de propiedades siniestradas.

Iteración 5 – Semanas del 06/02/2012 al 21/02/2012

- Gestión Solicitudes de Envío: Se implementará la gestión de solicitudes de envío, se podrá cambiar el estado a enviada y luego indicar el estado en el que llegan los insumos.
- Costos: Se diseñaran altas, baja, modificación de costos y tipos de costos.
- Calculo de índices: Se implementará el cálculo del índice IDL e IGR.

Fase Testing

Iteración 1 – Semanas del 22/02/2012 al 01/03/2012

- Se relazará testing de integración.