Universidad de las Ciencias Informáticas

Facultad 2

"Sistema de Difusión de Información de Emergencias"

Trabajo de Diploma

Presentado en opción al título de Ingeniero en Ciencias Informáticas

Autor: Luz María Gutiérrez Feria

Tutores: Rammel Maestre Sánchez.

Wilson Alba Cal.

"Año 54 de la Revolución"

Ciudad de la Habana, Cuba.

Fecha de Junio de 2012.

Opinión del Tutor

Dedicatoria

A mi mamá, la que con tan solo amor, perseverancia, rectitud, confianza y tanto sacrificio ha logrado convertirme en la mujer y profesional que soy hoy. A ella, que no me ha dejado un instante sentirme sola a pesar de la distancia que nos ha separado estos 5 años, entre lágrimas y muchas más alegrías. A mi Macito, por habernos convertido en eso: un macito de amor. Para ti todo este esfuerzo y dedicación de quien te ama: tu María Mantilla (los ojitos de mamá).

A Yunán, por quererme tanto y creer en mí.

A mi papá, por su sacrificio.

A tía Modesta, Sheila y Julio, por dejarme ser su otra niña, por estar siempre conmigo y consentirme tanto. Por todo su apoyo y amor, a mí y a mi mamá, aquí está su ingeniera.

A Jeanne, que no necesita presentación, porque es mi eterno compañero. A él por estar siempre ahí para mí, por tolerarme en todos los aspectos, por su amor y paciencia.

Agradecimientos

A Fidel y a esta Revolución por permitirme lograr mis metas de ser una profesional.

A mi mamá, por estar siempre ahí siempre para todo lo que necesito.

A Rammel, mi tutor, por su innumerable apoyo, tiempo, dedicación y confianza en todo este recorrido, a él no se la dedico porque forma parte de este logro, ha sido mi compañero de tesis estos 6 meses de trabajo. Gracias.

A Adys, por ayudarme tanto con sus conocimientos, y estar ahí siempre cada vez que la necesitaba.

A todos mis compañeros del proyecto, Ráiner, Yadier, Yohanna, Dany, Yuliani, Yamila, que tanto apoyo y preocupación recibí de ellos.

A todas las niñas del equipo de voleibol UCI femenino y a mi entrenador por todos los momentos buenos que pasamos juntos y por formar parte de mi vida todo este tiempo, que lograron la estabilidad en mí con la tensión de la tesis. Y a mi eterna compañera y campeona que no podía faltar, Isel, por su amistad y apoyo.

A todos gracias.

Resumen

El presente trabajo constituye una propuesta para solucionar los problemas existentes en la gestión, dentro de los Centros de Gestión de Emergencias 171, de la publicación de información de emergencia en Internet de manera que contribuya a mantener a los ciudadanos actualizados en tiempo real. Para la construcción del sistema desarrollado se seleccionó como metodología de desarrollo RUP (Proceso Unificado de Desarrollo por sus siglas en inglés), NetBeans 7.1 como IDE (Entorno de Desarrollo Integrado por sus siglas en inglés), el gestor de base de datos Oracle10g y Apache Tomcat como servidor de aplicaciones; definiendo también, como framework base de la solución propuesta Grails en su versión 2.0.3. Se concluyó que después del análisis realizado a los sistemas destinados a la difusión de información de emergencia a través de las tecnologías de la información y las telecomunicaciones a nivel mundial, los mismos no se ajustan a las necesidades de los Centros de Gestión a Emergencias 171. Además, el uso de Grails como framework principal, facilitó el desarrollo de la aplicación, reduciendo tiempo y esfuerzo por parte del equipo de desarrollo, logrando una mayor productividad, mientras que los casos de pruebas ejecutados, arrojaron resultados satisfactorios, demostrando que la aplicación desarrollada realiza todas las funcionalidades previstas. El sistema implementado cuenta con las funcionalidades necesarias para gestionar, dentro de los Centros de Gestión de Emergencias 171, la publicación de información de emergencia en Internet, contribuyendo a mantener a los ciudadanos actualizados en tiempo real.

Palabras claves: sistema web, Grails, gestión de emergencia.

Índice

Introducción

La prevención de la violencia y la seguridad ciudadana han tomado gran importancia en los últimos años sumándose a las preocupaciones fundamentales de los gobiernos de varios países. Sin importar el país donde viva, sentirse seguro es un derecho de cada ciudadano.

La seguridad ciudadana es aquella situación de normalidad en la que se desenvuelven las personas, desarrollando actividades individuales y colectivas con ausencia de peligro o perturbaciones; siendo esta un bien común esencial para el desarrollo sostenible tanto de las personas como de la sociedad. Además, es aquella acción donde se involucran, para fines de la seguridad pública, tanto la acción política de la ciudadanía, como las actividades que por ley el Estado tiene que proporcionar. (Universidad de las Ciencias Informáticas, 2011)

La República Bolivariana de Venezuela, para garantizar el orden, la integridad y la buena convivencia dentro de su sociedad acudió a la implementación del Servicio de Atención a Emergencias 171 a nivel nacional, materializado en la inauguración de Centros de Gestión a Emergencias; siendo estos a su vez las entidades coordinadoras de los Órganos de Seguridad Ciudadana ante situaciones de emergencia. Los mismos, para la automatización de sus procesos, utilizan el Sistema de Gestión de Emergencia de Seguridad Ciudadana (SIGESC).

En la actualidad, diversos son los sistemas orientados a la difusión de información en Internet: redes sociales, blogs; que son altamente utilizados por los usuarios para comunicarse a diario intercambiando información entre sí. Los Centros de Atención a Emergencias no están exentos a este auge y han comenzado a aprovechar redes sociales como Facebook y Twitter, para difundir información de emergencia en Internet y así llegar a tiempo a la población.

Dentro de los procesos a desarrollar en el SIGESC, no está definida la difusión de información de emergencia. En los centros 171, estas funciones se hacen de manera descentralizada y sin un control adecuado del contenido de emergencia que se publica, pues en los casos en que se realiza, el director del centro o una persona designada, hace la labor de forma independiente sin seguir un procedimiento único y específico.

Actualmente, la realización de esta difusión constituye un proceso engorroso, debido a que hay que conectarse directamente a cada sitio a través del navegador, y escribir una y otra vez en cada página la información donde el centro quiera difundirla.

Ofrecer una comunicación de alerta en situaciones de emergencia es vital para reducir el número de víctimas. Llegar a este objetivo es todo un reto debido a la diversidad de los usuarios: personas con discapacidad, ancianos y niños, o simplemente otros grupos vulnerables. Propagar la información que constituye emergencia es un elemento de gran importancia, ya que los ciudadanos deben estar actualizados en tiempo real de cualquier situación: bloqueo de calles, posibles inundaciones, además de dar a conocer de la existencia del Servicio de Atención a Emergencias 171 en sí y el modo en que pueden cooperar con el mismo.

Se definió por tanto como **problema a resolver:** ¿Cómo gestionar dentro de los Centros de Gestión de Emergencias 171, la publicación de información de emergencia en los medios de difusión en Internet que contribuya a mantener a los ciudadanos actualizados en tiempo real?

Objeto de estudio: El desarrollo de software en los procesos de difusión de información de emergencia.

Campo de acción: Los sistemas de difusión de información de emergencia.

Objetivo: Desarrollar un sistema para gestionar, dentro de los Centros de Gestión de Emergencias 171, la publicación de información de emergencia en Internet que contribuya a mantener a los ciudadanos actualizados en tiempo real.

Tareas:

1. Estudio de los antecedentes del desarrollo de sistemas de difusión de información de emergencia.
2. Definición, modelación y descripción de los elementos del diseño para la posterior construcción de la solución.
3. Construcción del sistema de difusión de información de emergencia.
4. Realización de pruebas al sistema.

El presente trabajo de Diploma presenta la siguiente estructuración:

Capítulo 1. **Fundamentación teórica:** en el presente capítulo se brinda una panorámica de los conceptos asociados al problema. Se hace un análisis de los sistemas dedicados a la difusión de información, además del estudio del estado actual haciendo un estudio de las metodologías de desarrollo de software así como del lenguaje de modelado que pudieran ser adecuados para el desarrollo de la propuesta final.

Capítulo 2. **Descripción de la solución**: se describe la propuesta, representando el negocio a través de procesos. Se especifican los requisitos funcionales y no funcionales del sistema, los primeros recogidos en casos de uso. Se hace el análisis y diseño del mismo, relacionando las tecnologías y herramientas estudiadas y utilizadas en el posterior desarrollo de la aplicación, así como el modelo de los diagramas de clases del análisis. Se define la arquitectura y con ello se muestran los diagramas de clases de diseño y el modelo físico de la base de datos. Se muestran los diagramas de despliegue y de componentes como parte de la implementación del sistema y se relacionan además, los casos de pruebas realizados a la aplicación.

1 *Fundamentación teórica*

Capítulo I: Fundamentación teórica

En el presente capítulo se enuncian los conceptos relacionados con el objeto de estudio que deben esclarecerse para lograr un mejor entendimiento de la temática tratada. Se expone además el estudio realizado a los sistemas de difusión de información y de la metodología y lenguaje escogidos para modelar la solución.

1.1 Marco conceptual

1.1.1 Difusión de información

Difusión es la acción y efecto de difundir (propagar, divulgar o esparcir). El término, que procede del latín diffusĭo, hace referencia a la comunicación extendida de un mensaje. (Nieto Alfonso, 2008)

El tratadista venezolano entiende por difusión el envío de mensajes elaborados en códigos o lenguajes universalmente comprensibles, a la totalidad del universo perceptor disponible en una unidad geográfica, sociopolítica, cultural. (Pasquali, 1979)

La difusión de información se define como el proceso por el cual se transmite al usuario la información que necesita o en darle la posibilidad de obtenerla. (Cantillo, et al., 2011)

Se asumen los aspectos de las definiciones antes analizadas y se deduce que la difusión es la acción de propagar, divulgar o esparcir cualquier tipo de información a un usuario final que necesita la misma.

1.1.2 Formas de difusión

No hay una forma única de difusión, sino diferentes tipos de productos y servicios capaces de vehicular la información hacia los usuarios. Se pueden distinguir dos formas básicas de difusión, la difusión bajo demanda y la difusión documental. En cuanto a las vías de difusión pueden ser: papel impreso, tablones de anuncios, expositores, soporte magnético para consulta en ordenadores, páginas web, correo electrónico, difusión verbal (persona a persona, conferencias, cursos), medios audiovisuales (videos informativos). (Cantillo, et al., 2011)

Existen casos típicos de difusión, la publicidad comercial o la radiodifusión de régimen competitivo; de divulgación, el llamado "periodismo científico"; de diseminación, la

distribución de información científica entre una base de datos y la industria, o la entrega de una investigación a posibles centros de decisión. (Pasquali, 1979)

Tras la definición expuesta por Cantillo. G, Palmera. R y Román. I en su artículo, se concluye que las formas de difusión son todas aquellas maneras de expandir la información por la vía de difusión que se prefiera, de manera tal que lleguen lo menos distorsionadas posibles a los receptores.

1.1.3 Emergencia

Según la Organización Mundial de la Salud (O.M.S.) la definición de Urgencia es "la aparición fortuita (imprevisto o inesperado) en cualquier lugar o actividad de un problema de causa diversa y gravedad variable que genera la conciencia de una necesidad inminente de atención por parte del sujeto que lo sufre o de su familia". (Scribd, 2009)

Emergencia: cualquier suceso capaz de afectar el funcionamiento cotidiano de una comunidad, pudiendo generar víctimas o daños materiales, afectando la estructura social y económica de la comunidad involucrada y que puede ser atendido eficazmente con los recursos propios de los organismos de atención primaria o de emergencias de la localidad. (Telemedik, 2003)

1.1.4 Sistemas de difusión de información de emergencia

Tras la investigación y la relación de los conceptos antes enunciados, se concluye que los sistemas de difusión de información de emergencia se denotan como un conjunto de elementos organizados y relacionados que interactúan entre sí. Tienen como objetivo principal, difundir toda información que constituya emergencia. A modo de ejemplo se tienen algunos espacios de los periódicos, los noticieros, los programas informativos radiales y las páginas web que se dedican a difundir este tipo de información constantemente por la red.

1.2 Sistemas de notificación de emergencia

1.2.1 Servicio de alerta de emergencia 911

Para la difusión de emergencia el 911 opera solamente a través de llamadas telefónicas a los números registrados en su base de datos. Es un servicio que puede entregar

mensajes grabados a la vez a individuos y contestadores automáticos en vivo, así como a la especial de TTY/TDD[1]. (Chalá, et al., 2000)

1.2.2 RapidReach

Es una herramienta web para la notificación de emergencias que se basa en aumentar la velocidad y la exactitud del aviso del suceso, en situaciones críticas. El servicio es habilitado a través de una página web, recibiendo las notificaciones mediante cualquier dispositivo de voz o de texto habilitado, ya sea teléfonos fijos, móviles, dispositivos inalámbricos, SMS (servicio de mensajes cortos por sus siglas en inglés), fax, y correo electrónico. La aplicación soporta los idiomas: inglés, español, italiano, francés, alemán, sueco y polaco. El sistema está basado en Windows, para trabajar con Internet Explorer o un navegador compatible.

1.2.3 SEMA4A

Simple Alertas de Emergencia para todos, por sus siglas en inglés, es una propuesta de ontología[2] que centra su enfoque en adaptar automáticamente la notificación de alertar a diferentes tipos de usuarios (ancianos, discapacitados, otros grupos vulnerables), en función del tipo de tecnologías que puedan acceder, teniendo en cuenta el impacto del desastre en la comunicación y las infraestructuras de alertas de la forma más rápida y eficiente posible.

Hace un análisis profundo de conceptos y conocimientos relacionados con emergencias y tecnologías de los medios de comunicación, definiendo cómo se deben incluir grupos vulnerables de la sociedad a la hora de difundir información de emergencia.

1.2.3 AlertOC

Es un sistema público del Condado de Orange en California, Estados Unidos, de notificación masiva diseñado para mantener a las personas que viven o trabajan en el Condado de Orange al tanto de la información importante en casos de emergencia. AlertOC difunde la emergencia mediantes llamadas telefónicas a los teléfonos que el usuario cuando realiza su auto – subscripción, registra en el sitio, así como a su correo

[1] TTY / TDD transmisión en modo teletipo que facilita a las personas con discapacidad auditiva usar las líneas telefónicas.

[2] Las ontologías proporcionan un recurso semántico para describir la información relacionada con un dominio específico.

electrónico y a través de mensajes de texto. El sistema utiliza la base de datos del 911 para el contacto de las familias del Condado de Orange. (A. Malizia, cols., 2008)

1.2.4 Reachplus Alertas Avanzadas

Es un paquete de software que ha sido desarrollado teniendo en cuenta la necesidad de la comunicación instantánea o de difusión de información crítica o de emergencia en toda la organización. Con las alertas ReachPlus avanzadas, se pueden transmitir alertas instantáneas a todos o un grupo selecto de usuarios. (Organización Nacional de Protección Civil, 13 de noviembre del 2001)

El sistema está diseñado para hacerlo funcionar en un centro de trabajo, con un número reducido de usuarios; la información se envía a través de la red como un mensaje emergente en la pantalla del usuario, independientemente de lo que están haciendo.

1.2.5 EMCOM

El sistema EMCOM (Sistema Nacional de Notificación de Emergencia de Alerta, por sus siglas en inglés) permite controlar, monitorear, medir y presentar la información para la toma de decisiones mejor y más rápidas, a través de medios como: ordenador portátil inalámbrico, PDA (Asistente Digital Personal por sus siglas en inglés) inalámbrica o como mejor se le conoce: ordenador de bolcillo u organizador personal, e-Mail, teléfono celular. (National Emergency Alert Notification System, 1998-2005)

El sistema, funciona independientemente del sistema operativo, hardware o navegador de internet que use el usuario. Incorpora además las notificaciones por correo electrónico, mensajes de texto a localizadores, teléfonos celulares, PDA, fax, a través del teléfono y de radio enlaces. Los mensajes aparecerán como "pop-up" en los boletines de las pantallas del ordenador al que se notifique, o se entregarán a los localizadores o teléfonos celulares. La información dada a conocer puede incluir fotos y videos. El sistema incorpora el registro detallado y cronológico de todos los mensajes disponibles para el personal usuario autorizado en cualquier momento.

1.2.6 Resultado del estudio de los sistemas analizados.

Tras el estudio de las herramientas detalladas anteriormente se concluyó que las mismas presentan características similares que servirán de base para la implementación del sistema. Se puede destacar que difunden lo que creen que es necesario es decir, la

información que consideran que es emergencia. Suministran dicho contenido a la población que desea ser avisados en situaciones críticas es decir los usuarios registrados dentro del servicio; siempre utilizando diferentes vías de comunicación. Pero a pesar de esto, dichos sistemas son herramientas propietarias que pertenecen a las compañías que lo utilizan, por lo que no pueden ser utilizados por los centros 171, pero además, no permiten ver ni modificar su código fuente por lo que no es posible detallar su confección, unido a esto no cumplen con las necesidades, ni se ajustan a la infraestructura existente en los Centros 171 en Venezuela.

1.3 Centros de gestión de emergencias en Venezuela

En la República Bolivariana de Venezuela los centros encargados de la atención de las emergencias son conocidos como Centros 171, debido a que 171 es el número único de emergencia nacional que permite el acceso telefónico a estos centros por parte de la ciudadanía en cualquiera de los estados que cuentan con este servicio. Surgen por la necesidad de dar seguimiento y respuesta inmediata ante situaciones que quebranten el orden público, así como para la atención de solicitudes de emergencias por parte de los ciudadanos integrando y coordinando los cuerpos de seguridad competentes. Actualmente existen varios estados que poseen Centros 171, con el propósito de brindar atención a las situaciones de emergencias, que funcionan las 24 horas, dentro de los cuales se encuentran:

- Centro de Atención de Emergencias Aragua 171.
- Emergencias Bolívar 171.
- Sistema Integral de Emergencia de Táchira 171.
- Centro de atención 171 Portuguesa.
- Centro 171 Falcón.
- Servicio Autónomo Centro de Atención de Emergencias 171, Aragua.
- Centro Integral de Seguridad y Emergencias (C.I.S.E.) 171, Área Metropolitana.

Los Centros de Gestión de Emergencias 171 o Centros 171, pretenden:

- Fortalecer las acciones de coordinación entre los órganos de seguridad ciudadana y la unificación de criterios en la toma de decisiones.
- Disminuir el tiempo de respuesta a las demandas de emergencias formuladas por la población.

- Mantener un registro, en tiempo real e histórico, de información de hechos delictivos, emergencias y desastres.
- Medir la eficiencia y eficacia de los programas sociales en la disminución del delito dentro de las comunidades, y el uso de la información para el desarrollo de estrategias.

Estos centros carecen de un sistema de difusión de información de emergencia centralizado que pueda ser adaptado al funcionamiento de cualquier centro 171 y que permita hacer llegar a la población toda la información que constituya emergencia lo más rápido posible, evitando así la menos cantidad de daños posibles.

1.3.1 SIGESC

El Sistema de Gestión de Emergencia y Seguridad Ciudadana 171 (SIGESC 171) es un sistema automatizado para la gestión de emergencias, puesto en funcionamiento actualmente en los Centros 171 en la República Bolivariana de Venezuela. Diseñado para trabajar sobre el sistema operativo Windows XP y desarrollado con la plataforma de .NET, conformado por 9 aplicaciones de escritorio y una aplicación web que automatizan áreas dentro del centro 171 como: recepción de las llamadas de emergencia, despacho de las unidades para solventar las solicitudes de emergencia, y la supervisión de los procedimientos que se realizan en el centro.

1.4 Proceso de desarrollo de software

Un Proceso de Desarrollo de Software es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto. Tiene la misión de transformar los requerimientos del usuario en un producto de software. (Pressman, 2001)

1.4.1 Metodologías de desarrollo de software

El uso de una metodología en cualquier proyecto de desarrollo de software que controle su ciclo de vida, es sumamente importante. La misma debe ser capaz de mejorar la productividad de los desarrolladores, así como del cliente permitiendo la creación de mejores productos de software.

Una Metodología de Desarrollo de Software es un conjunto de técnicas, herramientas, procedimientos y documentos auxiliares que permite a los desarrolladores definir los elementos necesarios para la construcción de un nuevo producto. (Rational Software Corporation., 2003)

Actualmente existen dos corrientes en lo referido a los procesos de desarrollo, los llamados métodos pesados o rígidos, y los métodos ágiles o ligeros. El primer método es aquel que se centra especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán, proponiendo generar abundante documentación. Va dirigida a quipos de desarrollo grandes, donde el software a desarrollar es complejo. Entre estas se encuentran RUP (proceso unificado de desarrollo por sus siglas en inglés) y MSF (marco de soluciones de Microsoft por sus siglas en inglés). (Rational Software Corporation., 2003)

La otra filosofía es la de las metodologías ágiles, hacen especial hincapié en la interacción con el cliente, están más orientadas a la generación de código con ciclos muy cortos de desarrollo, contando con un equipo pequeño de desarrolladores. Tienen como base de sus resultados la comunicación con los usuarios involucrados en el proceso. Algunos de sus exponentes más destacados son XP (programación extrema, por sus siglas en inglés) y RAD (desarrollo rápido de aplicaciones por sus siglas en inglés).

La metodología definida para que guíe el desarrollo del Sistema de Difusión de Información de Emergencia será RUP, utilizada también por el equipo de desarrollo del SIGESC, donde forma parte el sistema propuesto, puesto que es necesario documentar a profundidad el proceso de desarrollo para posteriores versiones, actualizaciones, así como para el estudio y capacitación de los miembros del equipo que está en constante cambio

1.4.1.1 El Proceso Unificado de Desarrollo (RUP)

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, es una metodología de desarrollo de software orientada a objetos, basada en UML (lenguaje de modelado unificado por sus siglas en inglés) y que se clasifica como pesada o robusta.

Los aspectos definitorios y a la vez que convierten en único al Proceso Unificado, se resumen en tres fases: dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

- **Dirigido por casos de uso:** un caso de uso es el conjunto de acciones que debe realizar un sistema para dar un resultado de valor a un determinado usuario; se capta cuando se modela el proceso del negocio y se representa a través de los requerimientos. A partir de aquí todos los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

- **Centrado en la arquitectura:** la arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

- **Iterativo e incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Con su culminación se obtiene un producto con un determinado nivel, que irá creciendo incrementalmente en cada iteración.

1.4.2 Lenguaje de modelado

Unified Modeling Language (UML, Lenguaje Unificado de Modelado) es el lenguaje más conocido y utilizado en el modelado de sistemas de software. UML al ser un lenguaje gráfico, nos permite visualizar, especificar, construir y documentar el sistema que se propone desarrollar. Ofrece un estándar para detallar un plano del sistema (modelo), incluyendo características conceptuales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguaje de programación, esquemas de bases de datos y componentes reutilizables. (Visual Paradigm International, Ltd., 2010)

Se puede emplear en el desarrollo de un software como soporte a una metodología como RUP, pero no se especifica que metodología o proceso se debe usar.

1.4.2.1 Notación para el Modelado de negocio

Para realizar el modelado de negocio se pueden emplear técnicas y notaciones que ayudan a conocer los objetivos del negocio y plasmarlos en un modelo.

Integrated Definition Methods (IDEF) es una familia de lenguajes de modelado con una amplia gama de usos como el modelado funcional, simulación, análisis orientado a objetos hasta el diseño y adquisición de conocimientos.

De la familia IDEF el más conocido y utilizado es IDEF0 ya que *"es una técnica de modelado común para el análisis, desarrollo, reingeniería, y la integración de los sistemas de información, procesos de negocio, para el análisis de ingeniería de software. IDEF0 se utiliza para mostrar el flujo de datos, sistema de control, y el flujo funcional de los procesos de ciclo de vida."*[3]

Utilizando IDEF0 se puede modelar los procesos del negocio desde el nivel de más alto de abstracción hasta el nivel de detalle que se desee, producto de la estructura jerárquica que posee para representar un proceso. Facilita la comunicación y captura de información y permite analizar, documentar y mejorar los procesos.

La notación IDEF0 se compone de actividades, entradas, salidas, mecanismos de control y sujetos.

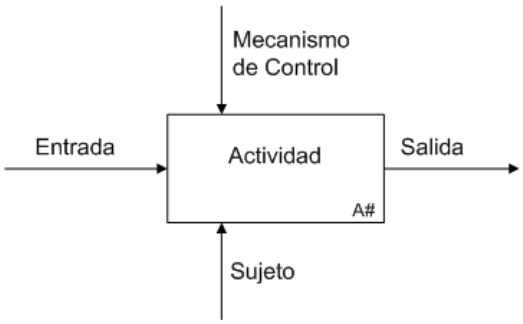


Figura 1. Formato Caja de IDEF 0

Cada actividad se representa con un rectángulo cerrado y en la esquina inferior derecha se pone el número de la actividad para seguir un orden. Las actividades deben tener obligatoriamente Entradas y Salidas y de manera opcional Mecanismos de Control y Sujetos que realizan la actividad.

[3] Adaptado de (Department of Defense. System Management College., 2001 p. 51).

La entrada se representa como una conexión que entra a la actividad por la izquierda siendo lo que inicia la actividad. La salida se representa como una conexión que sale de la actividad por la derecha y es el resultado que se obtiene después de haber realizado la actividad.

Un mecanismo de control es una conexión que entra a la actividad por la parte superior y es el que indica las regulaciones que determinan si una actividad se realiza o no.

Un sujeto se representa por una conexión que entra a la actividad por la parte inferior.

Dentro de las ventajas que ofrece IDEF0 se encuentra que es flexible ante cambios, explica los procesos más complejos de forma fácil, permite descomponer una actividad como un proceso a su vez, e identifica posibles procesos redundantes o defectuosos, y de esta manera organiza el negocio y lo informatiza.

1.4.3 Herramientas CASE

Las Herramientas de Ingeniería de Software Asistida por Computadoras (Computer Aided Software Engineering, CASE por sus siglas en inglés) son aplicaciones informáticas destinadas a incrementar la productividad y el control de calidad en cualquier proceso de elaboración de software, ya que transforman la actividad de desarrollar software en un proceso automatizado. Entre sus objetivos están: aumentar la calidad del software, mejorar el tiempo de desarrollo, costo y mantenimiento de los sistemas informáticos, ayudar a la planificación del proyecto, reutilización del software y documentación, generación de código, prueba de errores y gestión del proyecto[4]. Visual Paradigm para UML y Rational Rose son dos de las más importantes de estas herramientas.

Visual Paradigm

Visual Paradigm, es una herramienta CASE profesional, multiplataforma, que da soporte al modelado visual con UML 2.1 ofreciendo distintas perspectivas del sistema. Dotada de una buena cantidad de productos o módulos facilita el trabajo durante la confección de un software y garantiza además la calidad del producto final.

Entre sus principales características se pueden mencionar:

[4] Adaptado de (Microsoft Corporation, 2009 p. Microsoft .NET)

- **Multiplataforma:** Soportada en plataformas Java para Sistemas Operativos Windows, Linux y MacOS X.

- **Modelamiento de Bases de Datos:** Generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos, mapeos de objetos y relaciones, ingeniería inversa desde gestores de bases de datos.

- **Interoperabilidad:** Intercambia diagramas UML y modelos con otras herramientas. Soporta exportar e importar a XMI, XML y archivos Excel. Importa archivos de proyectos de Rational Rose. Integración con Microsoft Office Visio.

- **Integración con Entornos de Desarrollo:** Apoyo al ciclo de vida completo de desarrollo del software: análisis, diseño e implementación, en IDE como Microsoft Visual Studio, Eclipse, NetBeans, Sun ONE, Oracle JDeveloper, JBuilder y otros.

- **Modelamiento de los Requisitos:** Captura de requisitos con diagrama de requisitos, modelamiento de casos de uso y análisis textual.

- **Modelado de Procesos de Negocio:** Visualiza, comprende y mejora los procesos de negocio con la herramienta más completa de notación de modelaje de procesos de negocio.

- **Colaboración de Equipo:** Realiza el modelado en colaboración y simultáneamente con el Visual Paradigm TeamWork Server, CSV y Subversion.

- **Ingeniería de Código:** Permite generación de código e ingeniería inversa en lenguajes como Java, C++, CORBA, IDL, PHP, XML Schema, Ada, Python, C#, VB .NET, ODL, Flash ActionScript, Delphi, Perl y Rugby. También permite ingeniería inversa desde clases Java, .dll y .exe de .NET, JDBC y ficheros de mapeo de Hibernate.

- **Generación de Documentación:** Comparte y genera los diagramas y diseños en formatos como PDF, HTML y Microsoft Word.

- **Editor de Detalles de Casos de Uso:** Entorno todo en uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.

Rational Rose

Rational Rose es la herramienta CASE de modelado visual para el análisis de diseño de sistemas orientados a objetos que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML 1.1 como lenguaje de modelado visual. Se utiliza para modelar el sistema antes de construirlo cubriendo todo el ciclo de vida del software.

Entre sus características fundamentales Rational Rose incluye: permite la generación de código a partir de un diseño en UML en lenguajes como C++, Visual Basic, Java y Ada; incluye generación de reportes entre otros. Soporta realizar ingeniería inversa por lo que se puede obtener un diseño a partir del código de un programa. Provee UML para modelamiento de bases de datos con representación en modelos lógico y físicos, incluyendo además que soporta sistema operativo Windows, integración con IDE de plataforma Java, Microsoft Visual Studio, Borland JBuilder y Sun Forte.

Selección de la Herramienta CASE

Se decide utilizar el Visual Paradigm, teniendo en cuenta que es una herramienta multiplataforma, dominada casi en su totalidad por el equipo de desarrollo, posibilitando esto facilidades en su uso y aprovechamiento del tiempo. Es además una herramienta amigable, que posibilita realizar las especificaciones de los casos de uso y generar documentación sin necesidad de utilizar herramientas externas. Es muy fácil de usar en la creación de todo tipo de diagramas UML, para los que dispone de un número considerable de estereotipos que permiten un mayor entendimiento de los mismos.

1.5 Conclusiones del capítulo

1. Tras el estudio de los sistemas destinados a difundir información de emergencia, los mismos no satisfacen las necesidades de los centros 171 en Venezuela.
2. Se seleccionó RUP como metodología que guiará el desarrollo del sistema, aprovechando la abundante documentación que genera que servirá para posteriores versiones de la aplicación, con Visual Paradigm como herramienta CASE para todos los diagramas generados en el desarrollo, utilizando UML como lenguaje de modelado.

3. Para realizar el modelado de negocio se escogió IDEF0 para así lograr un mejor entendimiento de los procesos.

2 *Descripción de la solución*

Capítulo II: Descripción de la solución

Tras haberse seleccionado RUP como metodología para el desarrollo del software, el presente capítulo se estructura siguiendo los principales flujos de trabajo de dicha metodología: modelación del negocio, requerimientos, análisis y diseño, implementación y prueba, describiendo en todo su conjunto la propuesta de la solución.

2.1 Modelación del negocio

La modelación de los procesos del negocio se realizará mediante la notación IDEF0. Los que fueron identificados se muestran en el siguiente diagrama:

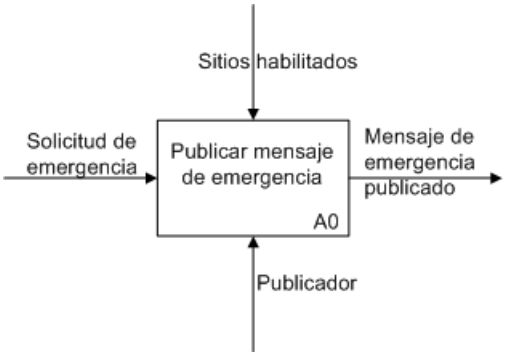


Figura 2. Diagrama IDEF0 Principal de procesos del negocio

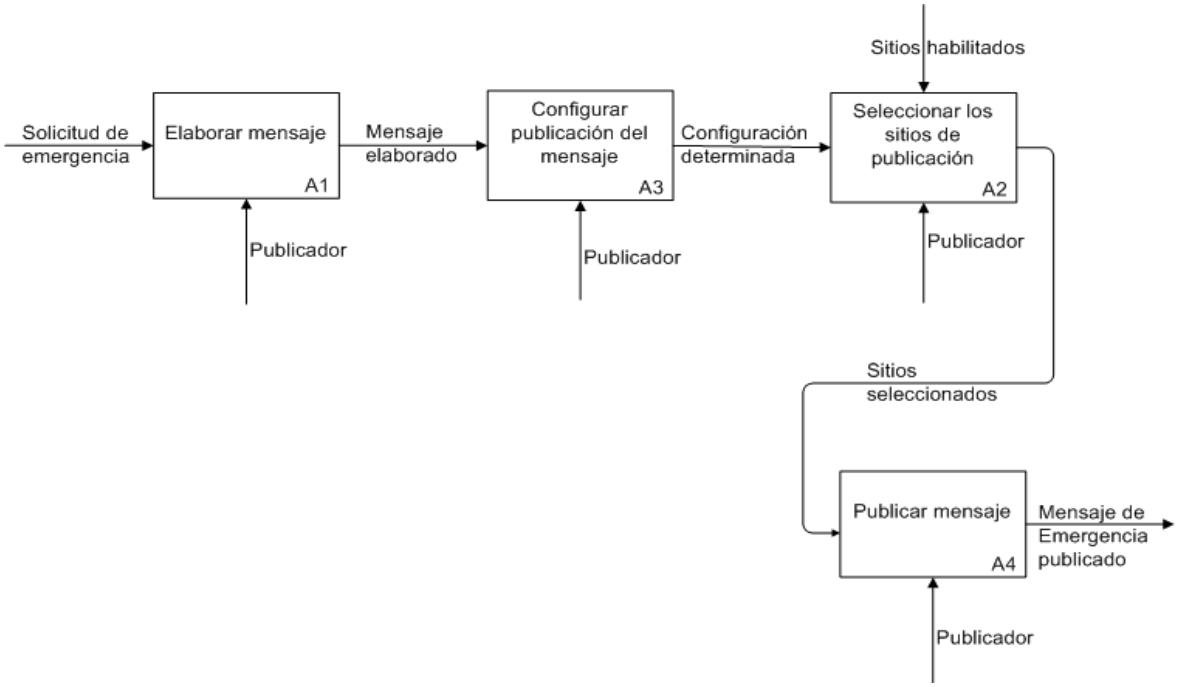


Figura 3. Diagrama del proceso de negocio A0

Tabla 1. Sujeto del proceso de negocio A0

Nombre	Justificación
Publicador	Persona encargada de realizar todos los procesos referentes a la publicación de los mensajes de emergencia.

2.1.1 Descripción del proceso del negocio

Publicar mensajes de emergencia

El publicador luego de elaborado el mensaje de emergencia, envía la información a los sitios donde se quiere difundir la misma.

Descripción de los procesos del Negocio Diagrama A0

- **Elaborar mensaje:** dadas las solicitudes de emergencia registradas en el centro, de las categorías de motivos[5] de solicitud existentes en la base de datos, el publicador elabora el mensaje de emergencia a publicar. El mensaje puede o no elaborarse a partir de las solicitudes; es el caso por ejemplo, de los mensajes orientativos a la población del uso del servicio de atención de emergencias 171 así como de la existencia de los centros de gestión de emergencia en sí.

- **Configurar publicación del mensaje:** luego de que el mensaje es elaborado, el mismo puede publicarse en el instante de su redacción o con una fecha, intervalo de fecha (fecha y hora de inicio de publicación y fecha y hora de fin de publicación) y frecuencia determinada. Estos aspectos son configurados para que el mensaje sea publicado cuando el publicador así lo determine.

- **Seleccionar los sitios de publicación:** para la publicación del mensaje, ya sea instantáneamente o luego de una configuración de la publicación, se hace

[5] Categoría de Motivo: Categoría que agrupa los diferentes tipos de motivo según una clasificación.
Motivo : Acción o hecho que da lugar a una situación de emergencia.
Ejemplo: Incendio (categoría) – Incendio forestal (motivo).

necesario saber en los sitios donde va a ser difundida la información, por ende, el publicador selecciona de los sitios habilitados, a dónde desea enviar el mensaje.

- **Publicar mensaje:** cuando el publicador termina de hacer todo el procedimiento de rutina y si no ha configurado la publicación del mensaje, es decir que quiere que el mismo se publique en ese instante, se dispone a publicar el mensaje, quedando el mismo publicado en los sitios seleccionados con el contenido elaborado, en otro caso, la configuración de la publicación del mensaje es almacenada para su posterior ejecución.

2.1.2 Reglas del negocio

Las reglas del negocio, son un conjunto de restricciones que deben cumplirse para el correcto funcionamiento del mismo. Define de acuerdo a los roles definidos el flujo de acción y las restricciones en las operaciones.

Reglas de estructura

Término: Usuario, Contenido de emergencia, Sitios de publicación

Modelo de datos:

- Un usuario de la aplicación es único en el sitio.
- Un usuario registrado debe tener asignado un rol.

Reglas de acción

Flujo:

- Un usuario se autentica en el sistema, el mismo verifica si tiene acceso, de ser cierto este ingresa al contenido del sitio. Si no, se le informa que el usuario o la contraseña esta incorrecta y por ende no tiene acceso al sistema.
- Cuando el usuario registrado entra al sitio, solamente se le permitirá acceder a los contenidos que en dependencia de su rol así lo permita.

Restricciones de operaciones:

- Solo el administrador del sistema puede habilitar/deshabilitar un sitio de difusión de información dentro del sistema.

Propuesta para la solución

El sistema permitirá como principal propósito publicar información de emergencia en los sitios de difusión de información en Internet, posibilitando además guardar el historial de las publicaciones con registro de fecha y hora de las mismas, usuario que lo envió, y los sitios donde fue dada a conocer la información. Brindará la posibilidad de programar la publicación de los mensajes con fecha y hora o rango de fechas determinadas. Se podrán habilitar y deshabilitar los sitios donde se quiera difundir la información, permitiendo añadir nuevos sitios de publicación, entre otras opciones.

2.2 Requerimientos

Con el conocimiento adquirido hasta el momento sobre los conceptos que rodean al objeto de estudio, se pueden analizar las características que debe tener el sistema para que se cumpla el objetivo planteado al inicio. Para ello se identifican los requisitos funcionales y no funcionales, modelando los requisitos funcionales en términos de casos de uso.

2.2.1 Requisitos funcionales

Una vez conocidos los procesos que interviene y los conceptos que rodean al objeto de estudio, se debe analizar: ¿qué debe hacer el sistema para que se cumpla el objetivo planteado al inicio de este trabajo?, para ello se enumeran, a través de requerimientos funcionales, las acciones que el sitio deberá ser capaz de realizar. Dentro de ellos se incluyen las operaciones que podrán ser ejecutadas por el usuario y las acciones ocultas que debe realizar el sistema.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. De acuerdo con los objetivos planteados este debe ser capaz de:

R1 Autenticar usuario

1.1 Verificar si el usuario tiene los permisos necesarios para ejecutar la aplicación

R2 Cerrar sesión

R3 Habilitar sitios de publicación

R4 Deshabilitar sitios de publicación

R5 Listar sitios de publicación

R6 Crear nuevos sitios de publicación

6.1 Registrar sitios con los siguientes datos:

- Nombre del sitio
- Url de acceso
- Habilitado/Deshabilitado
- Imagen de identificación

R7 Elaborar mensaje de emergencia

7.1 Registrar mensaje de emergencia con los siguientes datos:

- Fecha/Hora
- Contenido
- Usuario que envía
- Configuración de la publicación del mensaje

7.2 Seleccionar los sitios donde se publicará la información.

R8 Guardar la configuración de publicación del mensaje

8.1 Almacenar dentro del sistema los siguientes datos para su posterior uso:

- Fecha y hora de inicio
- Fecha y hora de fin
- Frecuencia de publicación

R9 Listar configuraciones registradas

9.1 Listar la configuración de los mensajes que están en proceso o pendientes de envío.

R10 Modificar configuración

10.1 Permitir modificar los siguientes datos:

- Fecha y hora de inicio
- Fecha y hora de fin
- Frecuencia de publicación

R11 Listar mensajes publicados

11.1 Realizar búsqueda de los mensajes publicados

11.2 Mostrar los siguientes datos de los detalles de los mensajes:

- Fecha/hora de publicación

- Usuario que lo envió
- Contenido del mensaje
- Datos de la configuración asociada

R12 Mostrar historial de publicación de los mensajes publicados

12.1 Realizar una búsqueda de la fecha, hora y sitio donde fue publicado

12.2 Mostrar los siguientes datos del historial del mensaje

- Sitio de publicación
- Fecha/hora en que se produjo el envío

R13 Publicar mensaje de emergencia

R14 Listar categorías de motivo de solicitud de emergencia

R14.1 Mostrar los siguientes datos de la categoría de motivo:

- Descripción

R15 Listar motivo de solicitud de emergencia

R15.1 Mostrar los siguientes datos del motivo:

- Descripción

R16 Seleccionar motivos de solicitud de emergencia

R17 Mostrar solicitud de emergencia

17.1 Mostrar los siguientes datos de la solicitud

- Dirección
- Descripción de la solicitud

R18 Mostrar detalles de solicitud de emergencia

18.1 Mostrar los siguientes datos de los detalles de la solicitud:

- Categoría y motivo al que está asociado
- Fecha de registro

- Dirección
- Descripción de la solicitud

R19 Editar tiempo de consulta de las solicitudes

R20 Configurar intervalos de tiempo de publicación

2.2.2 Requisitos no funcionales

Los requisitos **no funcionales** son propiedades o cualidades que el producto debe cumplir con el objetivo de lograr un producto atractivo, usable y confiable.

Tabla 2. Requisitos no funcionales

Categoría	Requisitos no funcionales del sistema
Usabilidad	1. La aplicación tiene que ser capaz de ofrecer facilidades de uso para un buen entendimiento y aceptación del producto por los usuarios finales.
Disponibilidad	1. Debe estar en funcionamiento las 24 horas del día, los 7 días de la semana, siendo este el horario de trabajo de los Centros 171.
Seguridad	1. Políticas de seguridad por usuarios y roles: el sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.
Portabilidad	1. La aplicación debe ser capaz de ejecutarse sobre diferentes sistemas operativos sin necesidad de modificar su código fuente.
Accesibilidad	1. Solo se podrá tener acceso dentro de la red del centro 171 donde está instalado.
Hardware	1. Tecnología Intel, 2 procesadores de 3.0 GHz, 2 GB de memoria RAM, 2 discos de 72 GB en RAID1.[6]
Software	1. Para el Cliente: Navegador Web (Mozilla Firefox recomendado). 2. Para el servidor: La máquina virtual de Java, servidor Apache Tomcat en su versión 6.0.32 y conexión a Internet.

[6] Modelo de despliegue del SIGESC v1.2. ALBET Ingeniería y Sistemas y/o Ministerio de Relaciones Interiores y Justicia. Fecha 24/04/2008. Referencia CT-SW-DR-031701.

2.2.3 Modelo de casos de uso del sistema

Utilizando las posibilidades que brinda el UML, se representarán los requisitos funcionales del sistema mediante un diagrama de casos de uso. Para ello hay que definir de acuerdo a lo planteado en los epígrafes anteriores, cuáles serían los actores que van a interactuar con el sistema y los casos de uso.

Tabla 3. Definición de actores del sistema

Actores	Justificación
Usuario	Representa el usuario encargado de elaborar, editar, listar, publicar los mensajes de emergencia, ver el historial de publicación de dichos mensajes, así como configurar el envío de los que no se publican en el instante que son elaborados.
Administrador	Representa la persona que administra y controla el comportamiento del sistema, dígase la gestión de los sitios de publicación, así como la modificación de la configuración de la publicación de los mensajes. Tiene derecho a realizar todas las operaciones definidas del sistema.

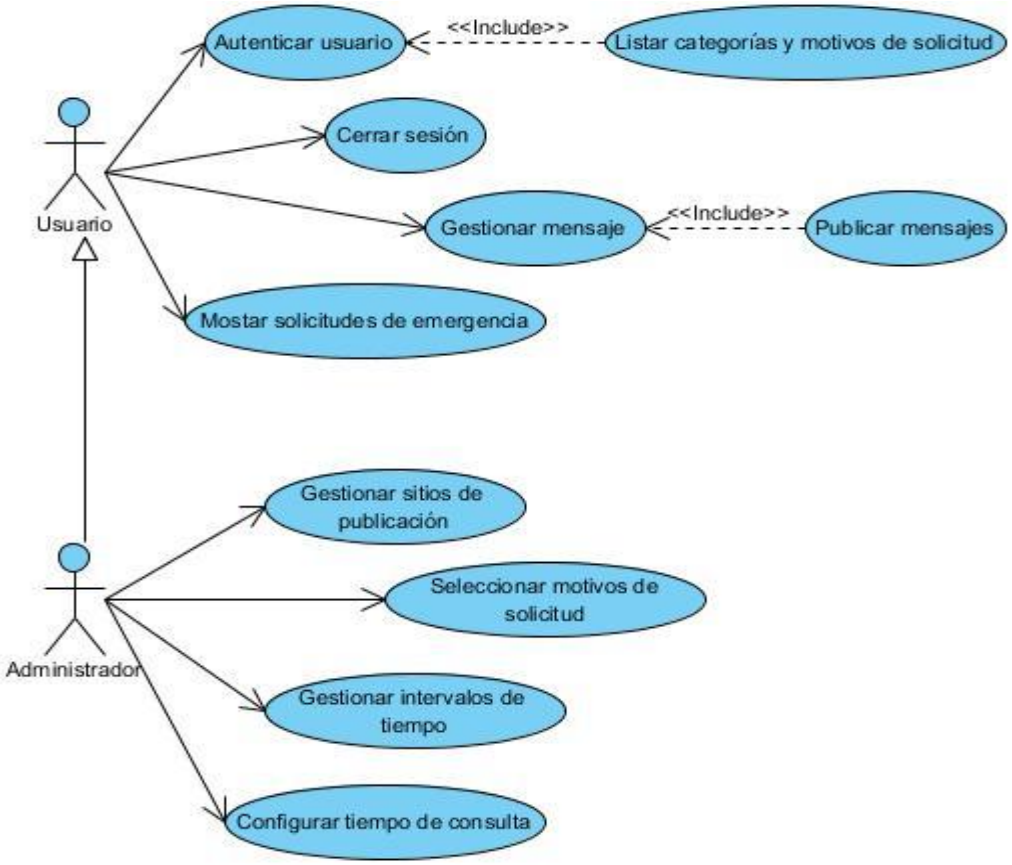


Figura 4. Diagrama de casos de uso del sistema

Las descripciones de los casos de uso del sistema se muestran en el Anexo1.

2.3 Análisis y Diseño

A continuación se describe cómo el sistema será realizado a partir de las funcionalidades previstas y las restricciones impuestas (requerimientos), indicando así, lo que se debe programar posteriormente.

Durante el análisis se analizan los requerimientos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo es conseguir una comprensión más precisa de los requerimientos y una descripción de los mismos que ayude a estructurar el sistema entero, incluyendo su arquitectura.

Luego de realizado el análisis se da paso al diseño en el que se modela el sistema y se encuentra su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se definieron. Una entrada esencial en esta etapa es el resultado del análisis, que proporciona una comprensión detallada de los requisitos.

2.3.1 Tecnologías y herramientas

A continuación se realiza la descripción de las tecnologías y herramientas estudiadas por el equipo de desarrollo, proceso que resulta fundamental cuando se desea realizar un software con la calidad necesaria y en el tiempo requerido.

2.3.1.1 Grails

Grails es un framework para desarrollo web, de código abierto y nativo de la Máquina Virtual de Java. Principios tales como Convención sobre Configuración (Convention Over Configuration), que busca disminuir el número de decisiones que un desarrollador necesita hacer, ganando así en simplicidad pero no perdiendo flexibilidad por ello, y No te Repitas (Don't Repeat Yourself), que promueve la reducción de la duplicación, forman parte de Grails. Ambos patrones en general proporcionan un entorno de desarrollo estandarizado y ocultan en gran parte detalles de configuración. (Smith, et al., 2009). Para el desarrollo del sistema se utiliza la versión 2.0.3.

Algunas de las características fundamentales de Grails son:

- *Alta productividad:* Grails tiene tres características que intentan incrementar su productividad comparándolo con los Framework Java tradicionales, la inexistencia de XML, es un entorno de desarrollo preparado para funcionar desde el primer momento y la funcionalidad disponible a través de métodos dinámicos.
- *Persistencia:* El modelo de datos en Grails se guarda en la base de datos utilizando GORM[7].

Grails trabaja por sus características sobre la Máquina Virtual Java (Java Virtual Machine, JVM por sus siglas en inglés), la cual es un componente fundamental de la plataforma Java capaz de ejecutar aplicaciones desarrolladas en este lenguaje. Está soportada en varios sistemas operativos como: Windows y Linux. Java está ampliamente distribuido en dispositivos móviles, servidores web y aplicaciones empresariales, aunque es menos común en aplicaciones de escritorio. Así, cuando se escribe una aplicación Java, se hace pensando que será ejecutada en una JVM en concreto. La gran ventaja de la JVM es la portabilidad que brinda al lenguaje pues luego de haber escrito el programa en Java este puede ser ejecutado en cualquier lugar que tenga instalada la JVM. La versión de la JVM utilizada para desarrollar el sistema es la 1.6 update 29.

[7] **G**rails **O**bject **R**elational **M**apping

También están presentes en Grails una serie de frameworks de código abierto tales como Hibernate para el mapeo objeto-relacional y para realizar consultas a la base de datos, SiteMesh para las plantillas y Spring para los demás manejos generales, todo esto junto al lenguaje de programación dinámico Groovy. Incluye además herramientas útiles para el desarrollo como: la Base de Datos de Consulta Estructurado (Hyperthreaded Structured Query Language Database) o HSQLDB, una base de datos en memoria para el desarrollo y pruebas, y Jetty que provee un servidor de aplicaciones para el desarrollo rápido y de carga automática. (Smith, et al., 2009)

Grails puede ser ampliado mediante plugins. Por lo dinámico que es, gracias a que aprovecha esta característica que le brinda Groovy, puede disminuir el ciclo de desarrollo ahorrando tiempo y agilizando el trabajo, se debe señalar que la instalación de Grails es muy fácil, pues todo lo que se necesita viene incluido en un paquete. Dentro de los plugins utilizados para el desarrollo de la aplicación están el Spring-Security para la seguridad del sistema y el plugin Quartz para la realización de tareas programadas, en este caso la configuración de la publicación de los mensajes de emergencia.

La arquitectura de Grails está conformada por 3 capas lógicas principales: Web Layer (capa web), Service Layer (capa de servicios), y Data Layer (capa de datos), donde cada capa está separada de la siguiente e interactúan mediante interfaces que definen funcionalidades que la misma debe brindar o también llamadas fachadas las cuales aseguran que el acoplamiento sea el más bajo posible y la abstracción del funcionamiento de la capa inferior, sea casi total. Cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Esta arquitectura en capas por su diseño proporciona la facilidad de modificar cada capa todo lo posible sin infligir daños o alteraciones a la capa inmediata.

Web Layer: En esta capa se encuentran las Vistas y la Lógica de Presentación, las cuales según la arquitectura que propone Grails estarán en los paquetes Controllers las clases controladoras y en View las Groovy Server Pages o más conocidas como GSP. En la Lógica de Presentación se manejará todo el flujo web utilizando la implementación del patrón Modelo Vista Controlador (MVC) que nos brinda Grails mediante Spring MVC, en el cual la lógica del negocio se separa de la presentación de la aplicación. Esto permite cambiar fácilmente el aspecto de la aplicación, sin modificar su comportamiento. La capa de presentación se compone principalmente de: modelo, vistas, controladores.

- **Controlador**: Un controlador de Grails es una clase responsable por el manejo de los pedidos provenientes de la aplicación. El controlador recibe la petición, realiza algún trabajo potencial con la misma y finalmente decide que sucederá a continuación.
- **Modelo:** Una de las actividades fundamentales llevadas a cabo por los controladores es obtener los datos que serán mostrados en la vista. El controlador puede recoger esta información directamente, delegarla a algún servicio u otro componente comprendido en la capa de acceso a datos, esta información es pasada a la vista en forma de un mapa u objeto de información. Dicho objeto representa el modelo.
- **Vista:** Grails utiliza para la interacción con el usuario la tecnología JSP. Pero basada en una implementación mediante GSP, que es una extensión de JSP y puede incluir Groovy. El mismo permite a los desarrolladores mezclar etiquetas de lenguajes de marcas tradicionales como HTML con código Java para producir vistas dinámicas. Las Vistas son los recursos que junto al modelo generado por los controladores le permiten al cliente visualizar la información.

Service Layer: En la capa de servicios se encapsula toda la lógica de la aplicación en fachadas de negocio que son utilizadas por los controladores en la capa de presentación y se exponen algunos procesos de negocio a través de interfaces de servicios. Sus clases radicarán según la arquitectura propuesta por Grails en el paquete Services.

Data Layer: En la capa de datos se manejan los objetos de acceso a datos abstrayéndolos del mecanismo de persistencia usado; a través de interfaces que exponen las operaciones de persistencia. Grails para evitar trabajar directamente con un gestor de base de datos y sus tablas y permitir trabajar con objetos en su lugar utiliza Hibernate 3 como una herramienta ORM[8]. Sin embargo dada la naturaleza dinámica de Grails y la adopción del convenio sobre la configuración, se crea sobre una versión superior de la implementación de Hibernate llamado Grails Mapeo Objeto-Relacional (GORM) que simplifica el trabajo con Hibernate y elimina cualquier configuración externa. (Nacho, 2009) (Judd, et al., 2008)

[8] **O**bject **R**elational **M**apping

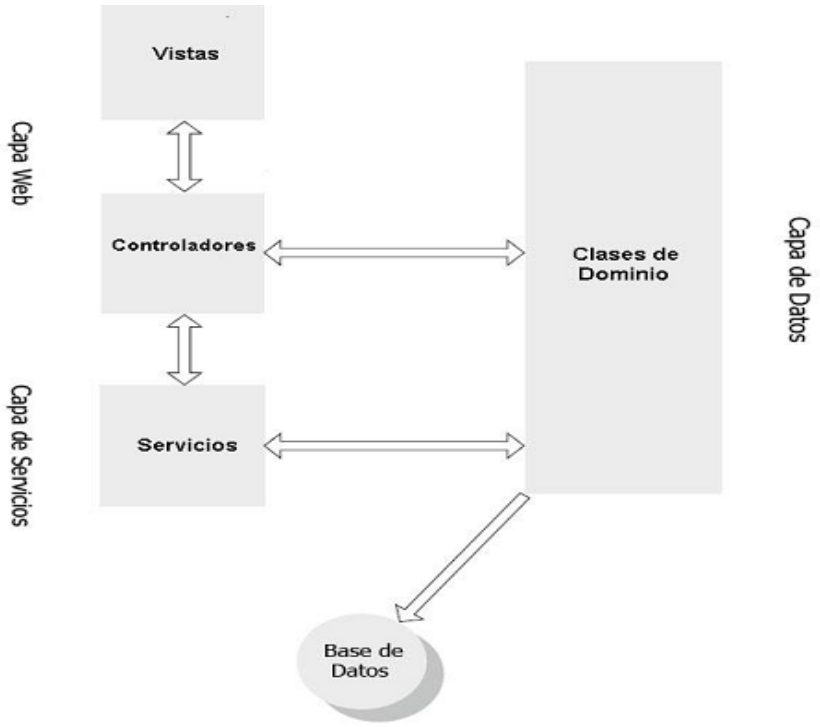


Figura 5. Arquitectura de Grails (Judd, et al., 2008)

2.3.1.2 Contenedor web

Es un servidor web con soporte para servlet y Java Server Page (JSP[9]), en la actualidad es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Es un software de código abierto multiplataforma y se encuentra bajo la Apache Software Licence. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la Máquina Virtual Java. Desde su origen ha evolucionado hasta convertirse en uno de los mejores servidores en términos de eficiencia, funcionalidad y velocidad. Tomcat es el servidor Web más utilizado a la hora de trabajar con Java en entornos web. (Apache Software Fundation, 1999-2012)La versión utilizada es la 6.0.32.

2.3.1.3 NetBeans

NetBeans es un entorno de desarrollo escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Es una aplicación de código abierto diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java. Dispone de soporte para el desarrollo de aplicaciones Web,

[9] **J**ava **S**erver **P**age

control de versiones y colaboración entre varias personas. (Rodríguez Guerra, y otros, 2009)

Características

- Editor de código sensible al contenido. Con soporte para autocompletar el código, coloreado de etiquetas y uso de abreviaturas para varios lenguajes de programación.
- Soporte para Java, Groovy, C, C++, XML y lenguajes HTML.
- Incluye control de versiones y compilación avanzada.
- Herramientas con asistentes para facilitar la escritura de código. (Luciano, 2008).
- Multiplataforma (Windows, GNU/Linux, Mac OS X y Solaris).
- Separa el diseño de software de la implementación con Modelado UML.

Se decide utilizar el NetBeans en su versión 7.1 como IDE, porque es multiplataforma, gratuito y de código abierto para desarrolladores de software. Tiene al alcance todas las herramientas necesarias para crear aplicaciones profesionales para entornos de escritorio en Java, tiene buen completamiento de código para Groovy como lenguaje de programación, manteniendo un aceptable funcionamiento en las computadoras personales (PC) que no tengan los mejores requerimientos de hardware como es el caso de la PC del equipo de desarrollo.

2.3.1.4 Sistema gestor de base de datos

Como sistema gestor de base de datos se seleccionó Oracle 10g debido a la robustez que posee, además por la seguridad y la integridad que garantiza de los datos al proyecto SIGESC 171.

Entre las características de Oracle se destaca su **escalabilidad** y **alta disponibilidad**, aportando un sistema de administración completo para gestionar todas las situaciones críticas de una base de datos, por ejemplo presenta: sistema de seguridad basado en backup y restauración de datos, alertas, usuarios, grupos y roles. (Microsoft Corporation, 2011)

Oracle soporta bases de datos de todos los tamaños, desde severas cantidades de bytes hasta gigabytes, además provee salvar con seguridad la información y asegurar de los errores vistos en el monitor y la información de acceso y uso. Establece un proceso entre

bases de datos del servidor y el cliente para la aplicación de programas, además de ser considerado altamente seguro. (Orabel, S., 2004)

En resumen las principales características que presenta este sistema son:

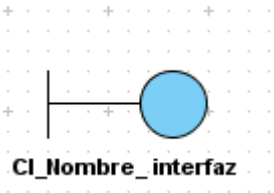
- Ayuda a administrar y almacenar grandes volúmenes de datos
- Se caracteriza por su estabilidad y escalabilidad.
- Es multiplataforma.
- Es un software privativo lo cual trae consigo algunas limitaciones para su uso.

2.3.2 Diagrama de clases del análisis

Las clases de análisis se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos.

Un diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. En este tipo de diagrama se representan las clases y sus relaciones. Ellos muestran una vista estática del sistema.

Tabla 4. Miembros del diagrama de clases del análisis

Nombre	Características	Representación
Entidad	Modelan información que poseen larga vida y que es a menudo persistente.	
Interfaz	Modelan la interacción entre el sistema.	

Control	Coordinan la realización de uno o unos pocos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.	

A continuación se ilustran los diagramas de clases del análisis del sistema.

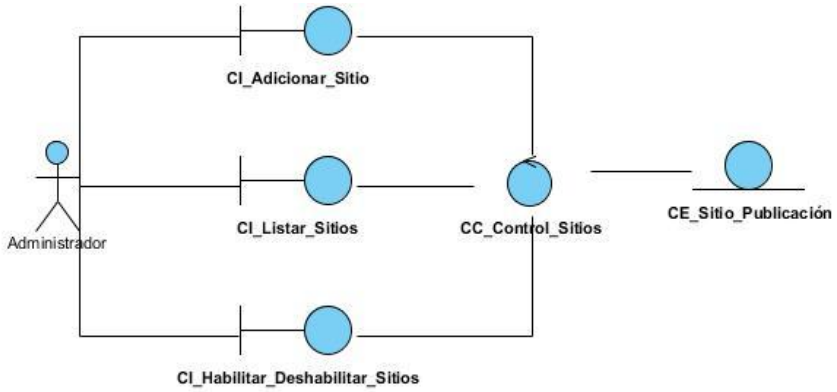


Figura 6. Diagrama de clases del análisis: Gestionar sitio

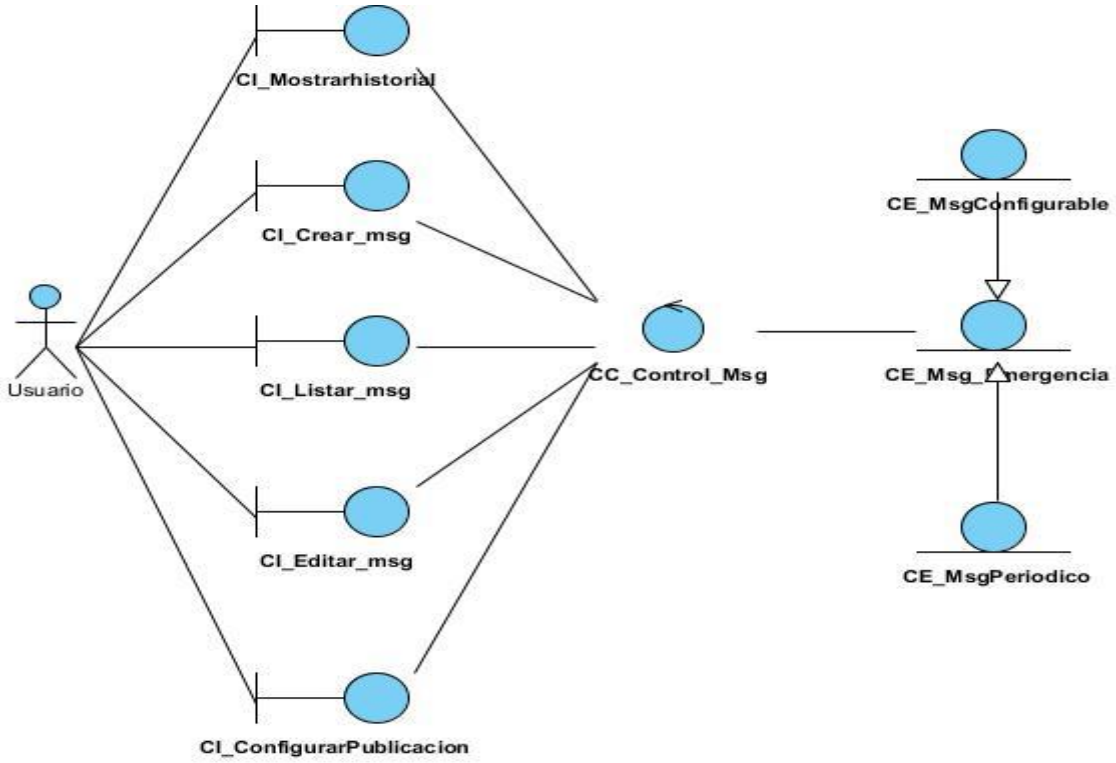


Figura 7. Diagrama de clases del análisis: Gestionar mensaje

Figura 8. Diagrama de clases del análisis: Publicar mensaje

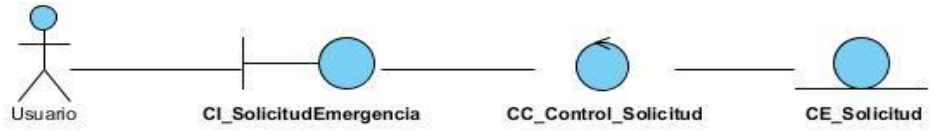


Figura 9. Diagrama de clases del análisis: Mostrar solicitudes de emergencia

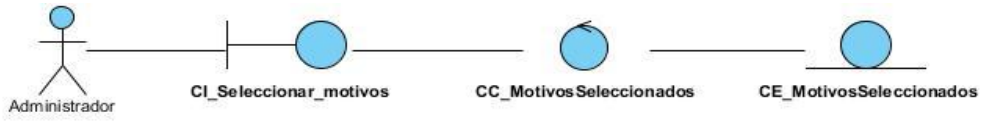


Figura 10. Diagrama de clases del análisis: Seleccionar motivo de solicitud

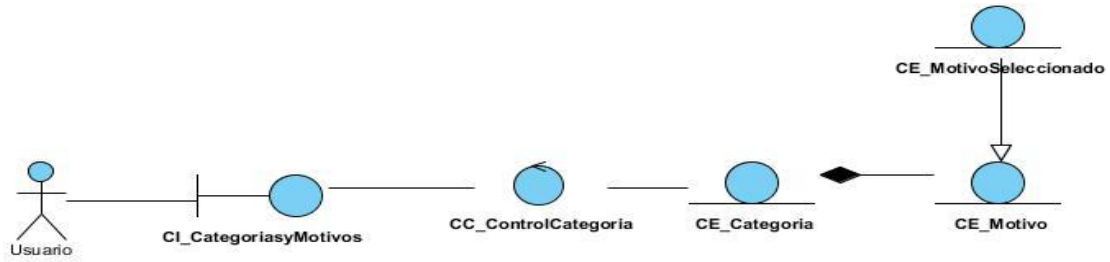


Figura 11. Diagrama de clases del análisis: Listar categorías y motivos de solicitud

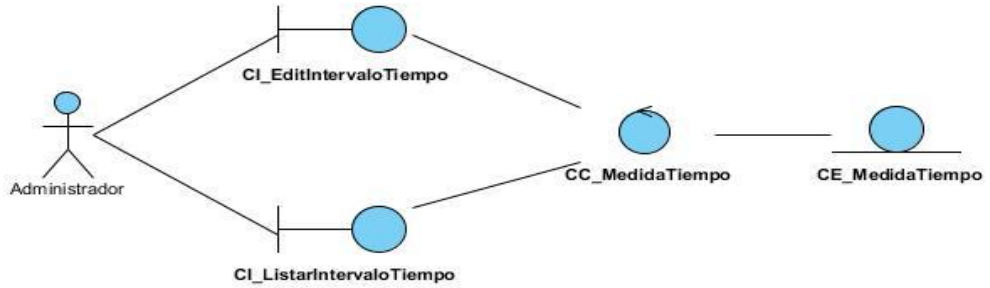


Figura 12. Diagrama de clases del análisis: Gestionar intervalos de tiempo

Figura 13. Diagrama de clases del análisis: Configurar rango de consulta

2.3.3 Arquitectura del sistema

La arquitectura de un software se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil, una guía de apoyo durante la fase de desarrollo de dicho software.

El sistema implementa la arquitectura cliente-servidor y está basado en la arquitectura propuesta por Grails que está conformada por 3 capas como se explicó anteriormente.

2.3.3.1 Capas lógicas del sistema

Capa Web

Los componentes de esta capa son responsables de mostrar al usuario el estado actual del modelo de datos y presentarle las distintas acciones disponibles. A esta capa pertenecen los archivos javascript (.js) que son los encargados de validar los campos y de esta forma velar porque los datos sean insertados correctamente; las vistas (.gsp) que son las que permiten al usuario interactuar directamente con el sistema; los controladores que son los que reciben las solicitudes del usuario (.groovy), realizan las operaciones de lógica de negocio sobre los modelos y decide qué vista será la que se mostrará a continuación; finalmente las hojas de estilo (.css) que son las que, a través de reglas, brindan el formato y el color a los componentes de la vista.

Capa de Datos

Esta capa contiene los componentes que representan y gestionan los datos manejados por la aplicación, es decir, los objetos encargados de leer y escribir en la base de datos. Las clases de dominio o entidades en las cuales va a persistir los datos de la aplicación pertenecen a esta capa. Las clases del dominio serán utilizadas por los servicios para obtener la información solicitada por el usuario o necesitada por el sistema.

Capa de Servicios

Esta capa contiene los servicios que son los componentes encargados de implementar la lógica de negocio de la aplicación. Los servicios manipulan las entidades a través de los métodos que poseen las mismas para insertar, modificar o eliminar los campos. Además GORM inyecta otros métodos en las clases persistentes que permiten realizar consultas a las tablas necesarias.

Grails genera componentes en cada una de las capas y es el entorno el que se encarga de conectar unos con otros y garantizar su buen funcionamiento.

2.3.3.2 Patrones de arquitectura

Los patrones de arquitectura ofrecen soluciones a problemas de arquitectura de software, dando una descripción de los elementos y el tipo de relación que tienen unido a un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones.

Cliente – Servidor

Esta arquitectura se encuentra dentro de la clasificación de estilo de llamada y retorno. El cliente y servidor generalmente están localizados en diferentes sistemas, o pueden encontrarse en el mismo medio. El cliente es la entidad que hace la petición por un servicio. El servidor es la entidad que provee el servicio correspondiente a la petición y el mismo debe procurar el resultado, el cual es retornado al cliente.

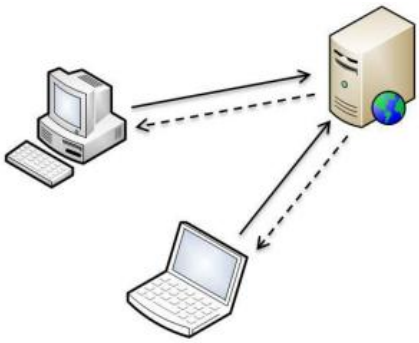


Figura 14. Arquitectura Cliente-Servidor

Modelo Vista Controlador (MVC)

Grails sigue un patrón de diseño muy popular en el mundo del desarrollo de aplicaciones web, el MVC que propone que los componentes del sistema se organicen en 3 capas distintas según su misión dentro del mismo.

- **Modelo o Capa de Datos:** Contiene los componentes que representan y gestionan los datos manejados por la aplicación, es decir, son los encargados de realizar las operaciones relacionadas con la base de datos.

- **Vistas o Capa de Presentación:** Los componentes de esta capa son los encargados de interactuar con el usuario, mostrarle las diversas acciones disponibles y el estado actual de los datos del sistema, esto se puede lograr a través de peticiones de actualización al modelo o a través de notificaciones de cambio que el modelo emite (eventos).

- **Controlador o Capa de Control:** Posee los componentes que reciben las órdenes de los usuarios, gestionan la aplicación de la lógica del negocio sobre el modelo de datos y determina la vista que se debe mostrar a continuación.

Cuando se dice que la capa de control "gestiona la aplicación de la lógica de negocio" hace referencia a que son los responsables de que esta se aplique, lo que no quiere decir que se implemente dicha lógica en las clases controladoras, pues esta se implementa en una cuarta capa, la cual se explica a continuación.

- **Capa de Servicios:** Contiene los componentes encargados de implementar la lógica de negocio correspondiente a la aplicación que se desarrolla.

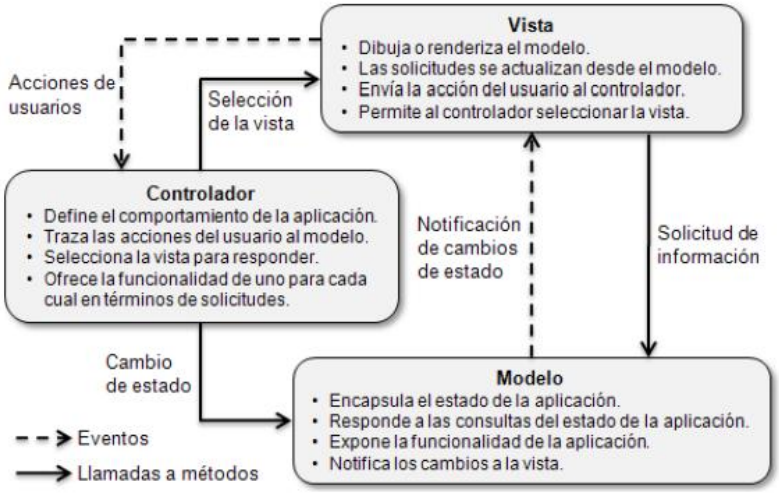


Figura 15. Esquema del patrón Modelo-Vista-Controlador (Buschmann, et al., 1996)

Inversión de control (IoC)

La inversión de control es otro patrón utilizado por Grails, según el cual las dependencias, de un componente no deben gestionarse desde el propio componente para que este solo contenga la lógica necesaria para hacer su trabajo. Cuando se crea un componente en la aplicación, Grails configura a Spring para controlar su ciclo de vida (cuándo se crea,

cuántas instancias mantiene vivas a la vez, cómo se destruyen, etc.) y sus dependencias (qué otros componentes necesita para desarrollar su trabajo y cómo seguirlos). Su objetivo principal es mantener los componente lo más sencillos posibles, incluyendo únicamente código que tenga relación con la lógica de negocio, siendo así la aplicación más fácil de comprender y mantener. (Smith, et al., 2009)

2.3.4 Diagramas de clases del diseño

Los diagramas de clases del diseño son la primera idea de cómo será la representación concreta de la aplicación. En cada uno de estos diagramas se representa tanto la vista con el formulario que ella contiene, así como también su respectiva clase JavaScript, además de la clase controlador que se relaciona con los servicios y las entidades necesarias para solucionar el caso de uso que se esté diseñando.

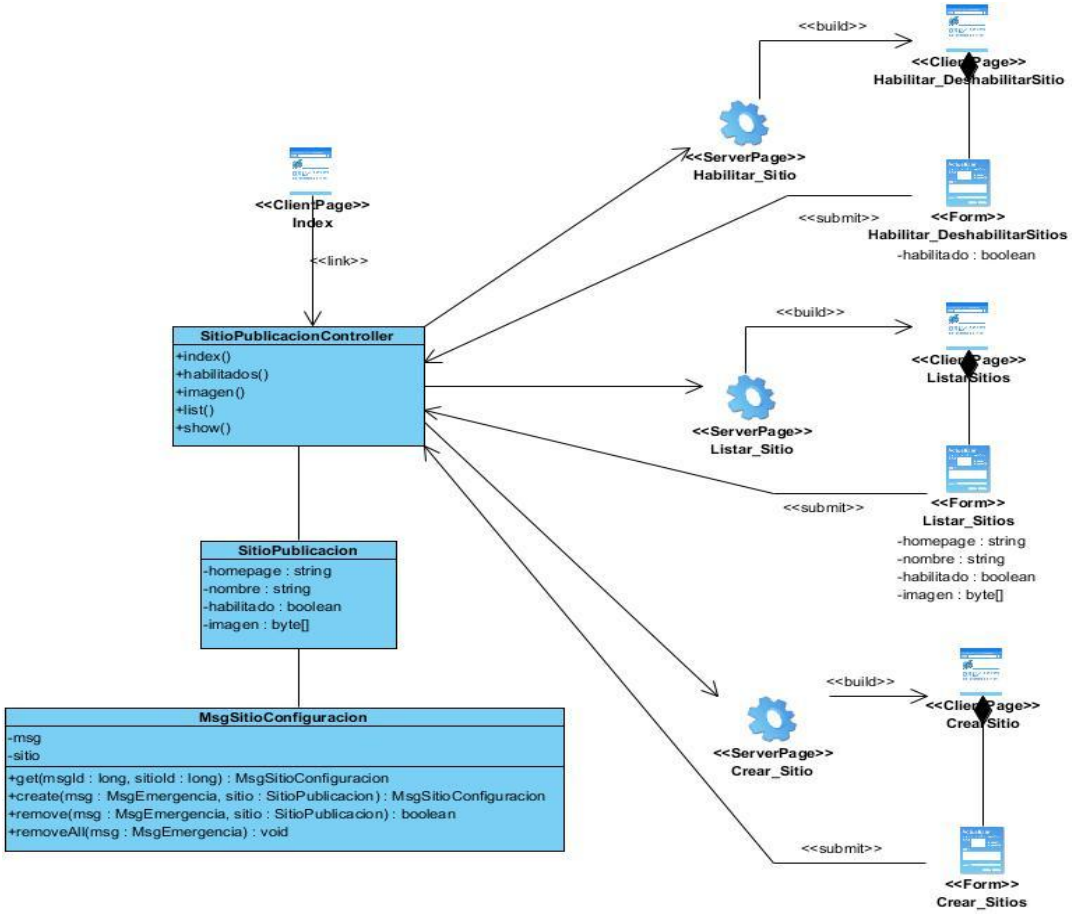


Figura 16. Diagrama de clases del diseño: Gestionar sitios de publicación

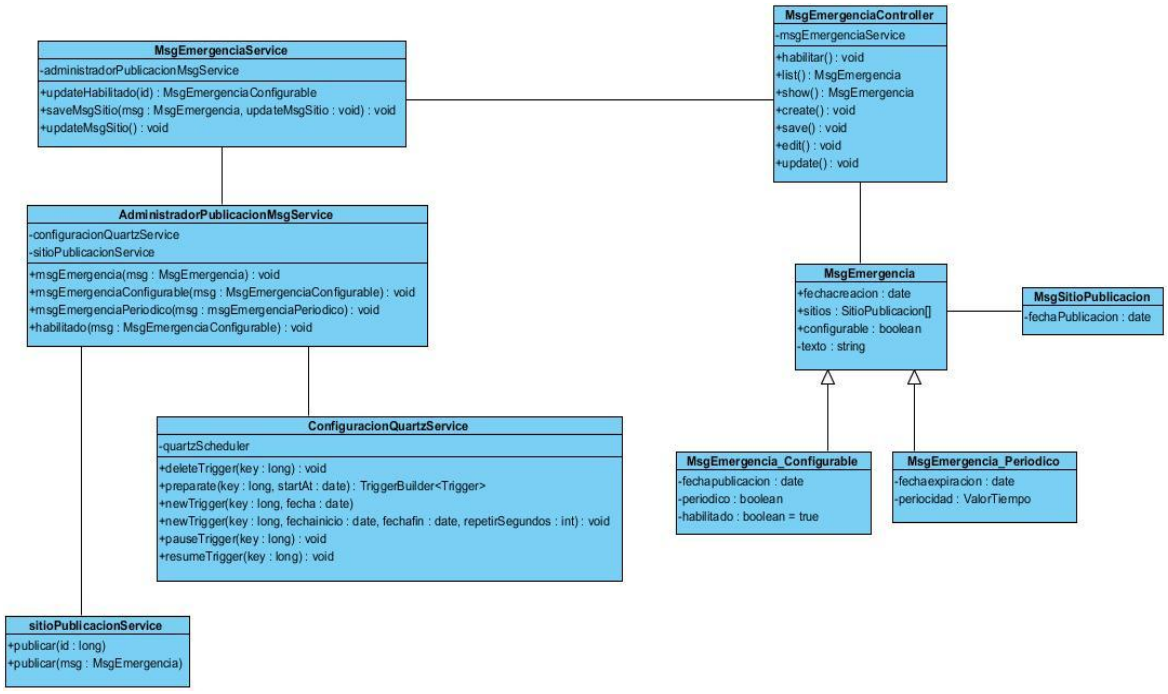


Figura 17. Diagrama de clases del diseño: Publicar mensaje

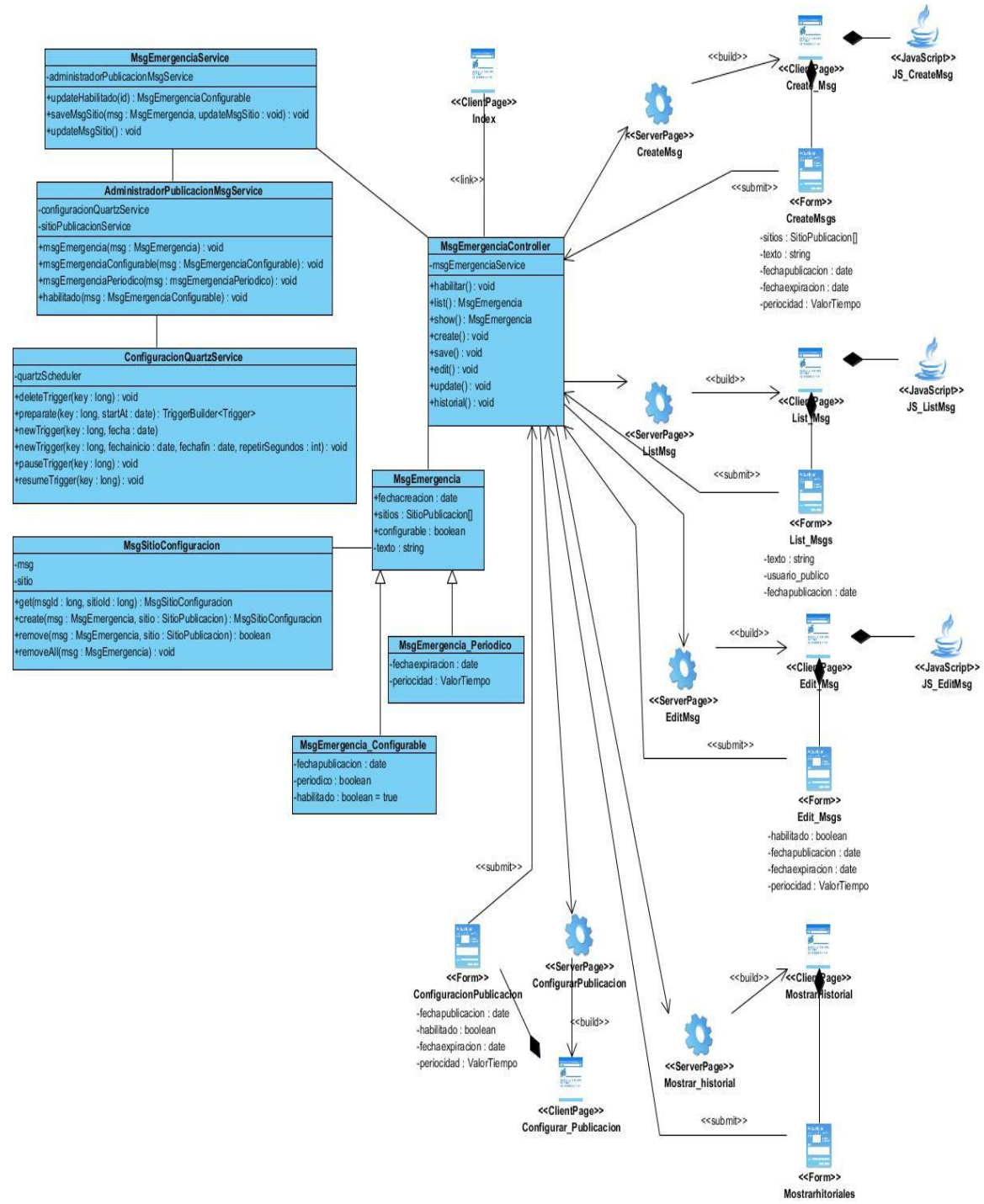


Figura 18. Diagrama de clases del diseño: Gestionar mensaje

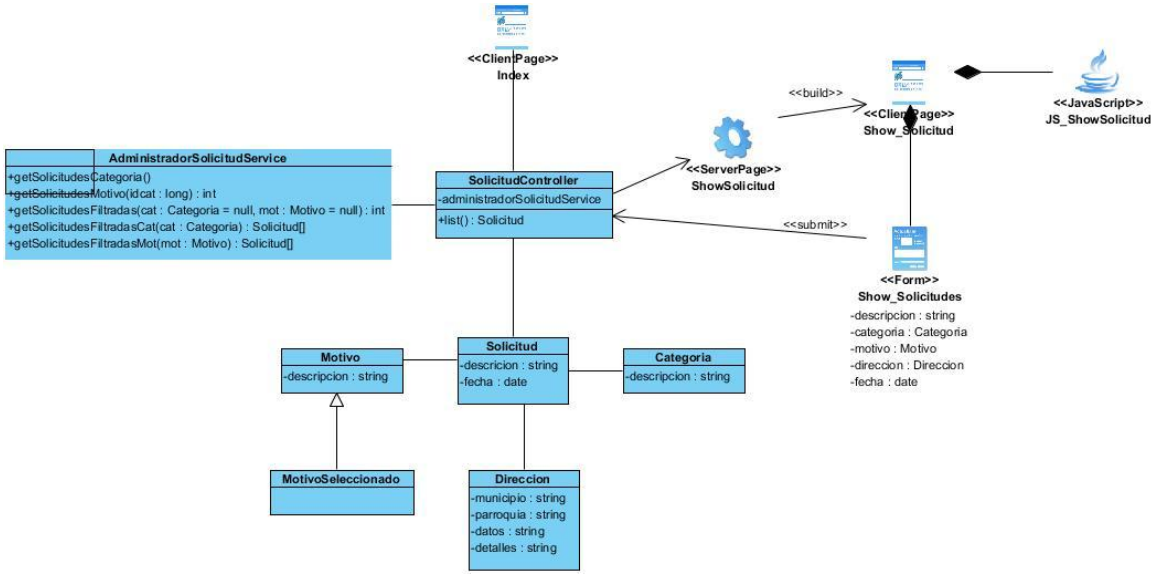


Figura 19. Diagrama de clases del diseño: Mostrar solicitudes de emergencia

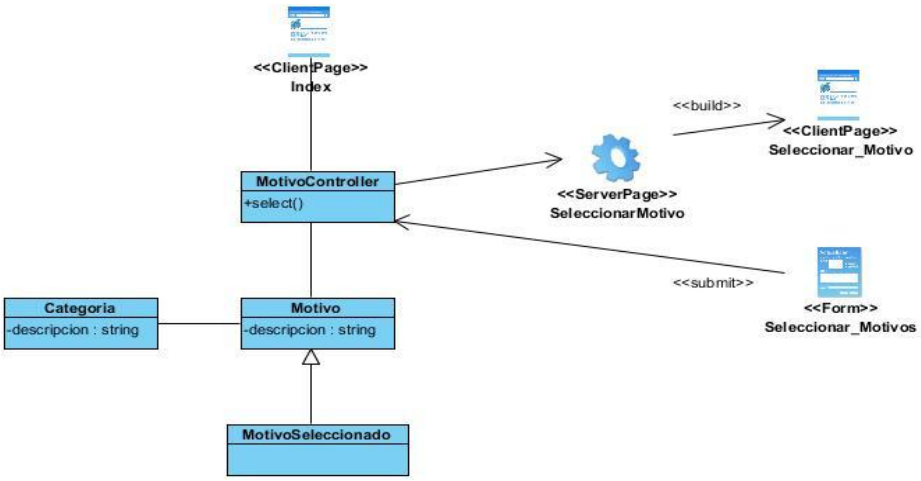


Figura 20. Diagrama de clases del diseño: Seleccionar motivos de solicitud

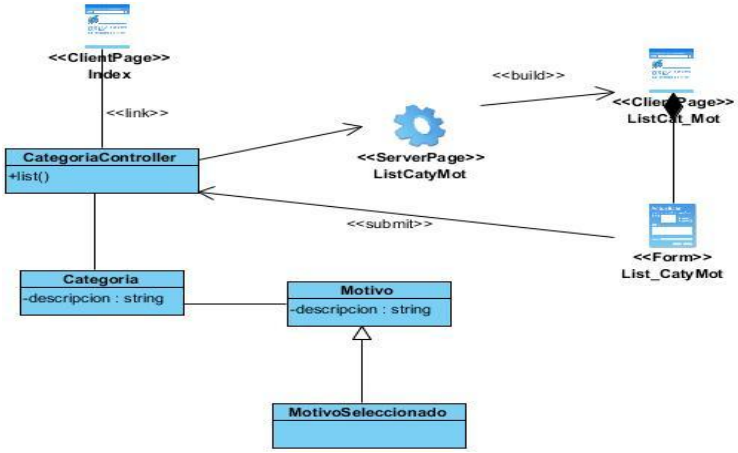


Figura 21. Diagrama de clases del diseño: Listar categorías y motivos de solicitud

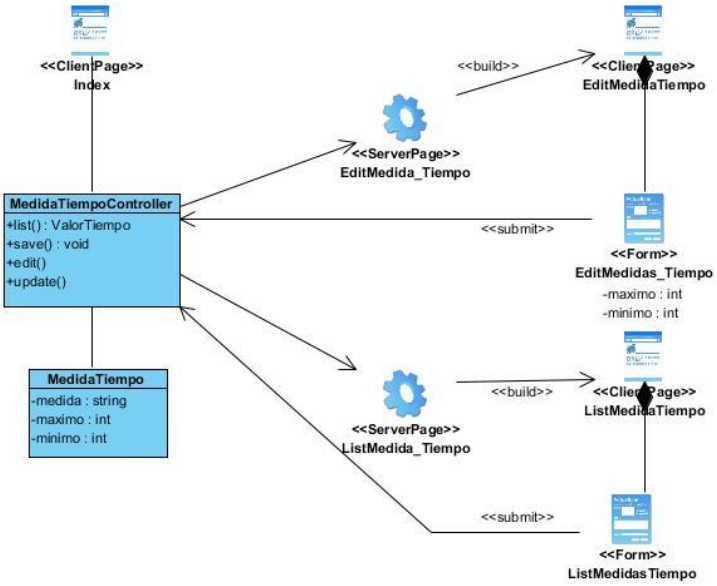


Figura 22. Diagrama de clases del diseño: Gestionar intervalos de tiempo

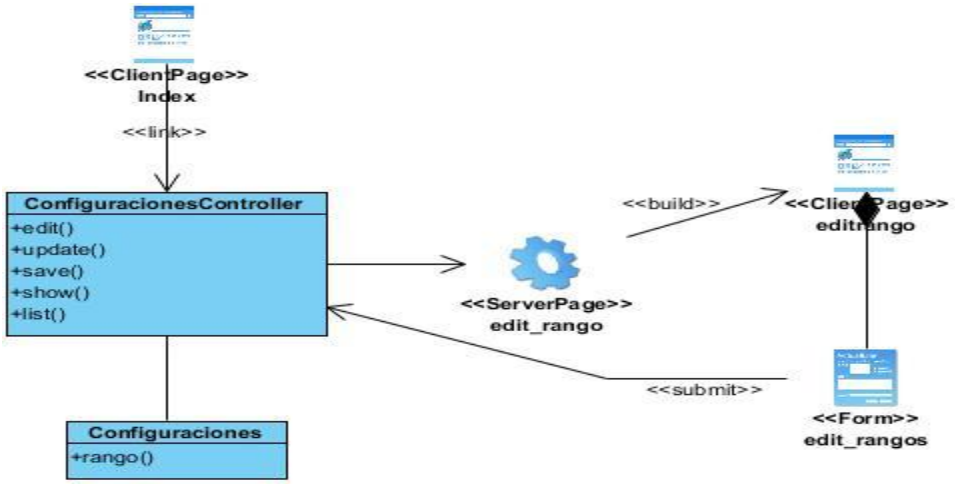


Figura 23. Diagrama de clases del diseño: Configurar tiempo de consulta

2.3.5 Diseño de la base de datos

A continuación se muestra el diagrama de Entidad-Relación definido para el modelado de la Base de Datos (BD) del sistema. Para la correcta implementación de la aplicación es necesario agregar entidades al modelo de BD del SIGESC, que respondan a las necesidades de las funcionalidades propuestas. Las entidades de color verde ya están en la BD pero para un mejor entendimiento es necesario incluirlas en el diagrama porque se relacionan con las nuevas entidades.

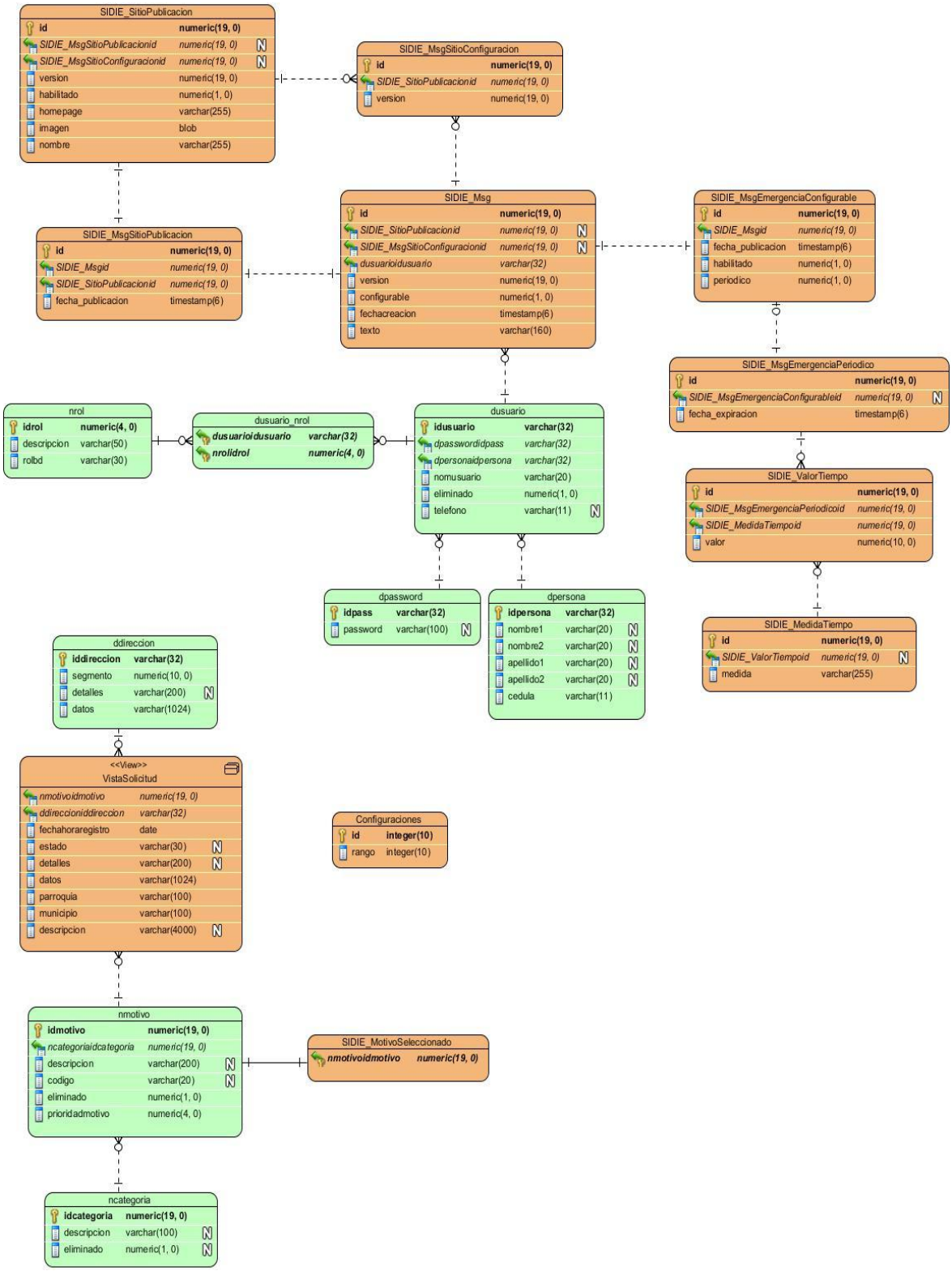


Figura 24. Modelo físico de la base de datos

2.4 Implementación

La implementación es la fase más esperada en un proceso de desarrollo de un producto de software, es donde se hacen realidad todas las ideas y artefactos que han sido modelados por el equipo de trabajo responsable de la solución; no constituye una etapa independiente y formalmente delimitada en el proceso, debido a que la metodología de desarrollo utilizada, permite que partes de la solución que conceptualmente estaban definidas y que se podrían ir prototipando de forma funcional, le dieran paso a la implementación mucho antes de que estos artefactos fueran modelados.

Durante este proceso se implementan las clases del diseño y se organizan en componentes, los cuales se ubican en nodos definiéndose la estructura en capas de la aplicación. Para la implementación del Sistema de Difusión de Información de Emergencia se realiza el diagrama de componentes y el de despliegue, los cuales se muestran a continuación.

2.4.1 Diagrama de despliegue

El modelo de despliegue contiene los nodos que conforman la topología de hardware sobre la que se ejecuta el sistema. Muestra las relaciones entre el hardware y el software en el sistema final y se representa como un grafo de nodos unidos por conexiones de comunicación. El modelo despliegue que se propone está formado por tres nodos. Un servidor de Base de Datos con Oracle 10g, corriendo en él la instancia que utilizará la aplicación (dbsigesc). El nodo Cliente que se conecta al sistema usando el protocolo http, la cual usará como sistema operativo Windows o Linux y como navegadores Internet Explorer o Mozilla (recomendado). Se necesitará además de un nodo portador del servidor web Apache Tomcat 6.0.32 donde estará ejecutándose el sistema (SIDIE).

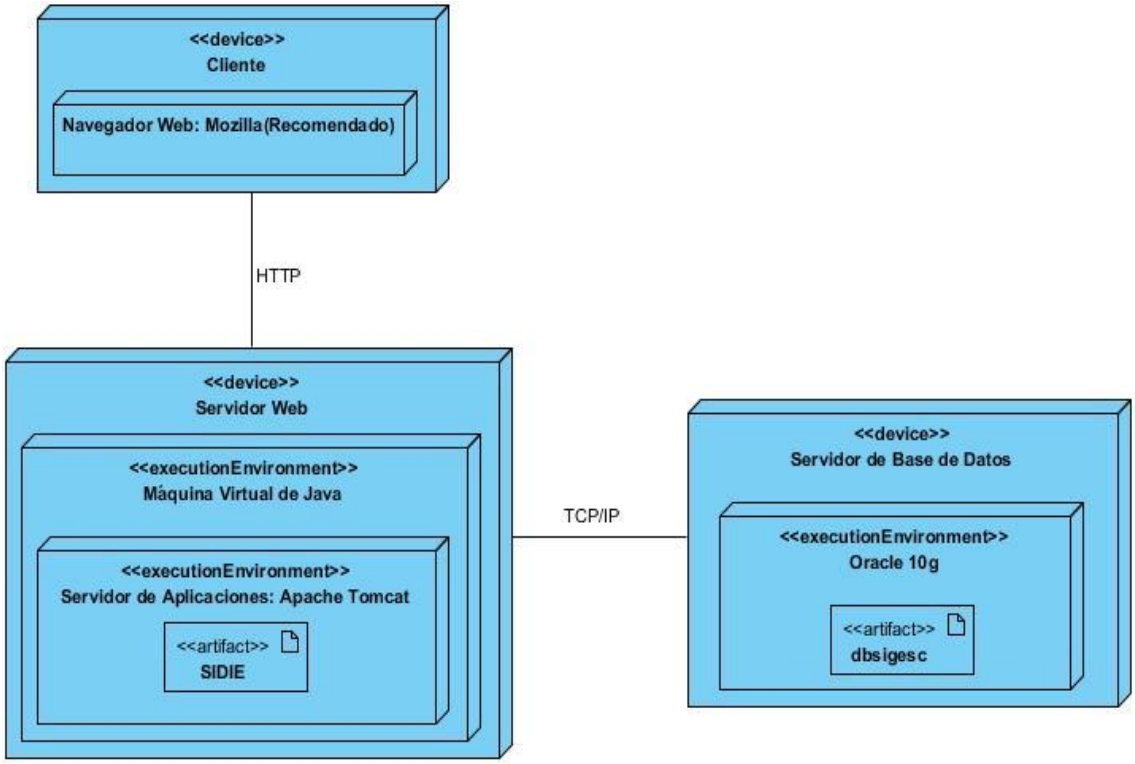


Figura 25. Diagrama de despliegue

2.4.2 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre los elementos del software, sean estos componentes de código fuente, binarios o ejecutables. Este diagrama muestra los elementos del software que constituyen una parte reusable, sus interfaces, y sus interrelaciones, en muchos aspectos se puede considerar que este tipo de representación es un diagrama de clases a gran escala. Normalmente los diagramas de componentes se utilizan para modelar código fuente, versiones ejecutables, bases de datos físicas, entre otros.

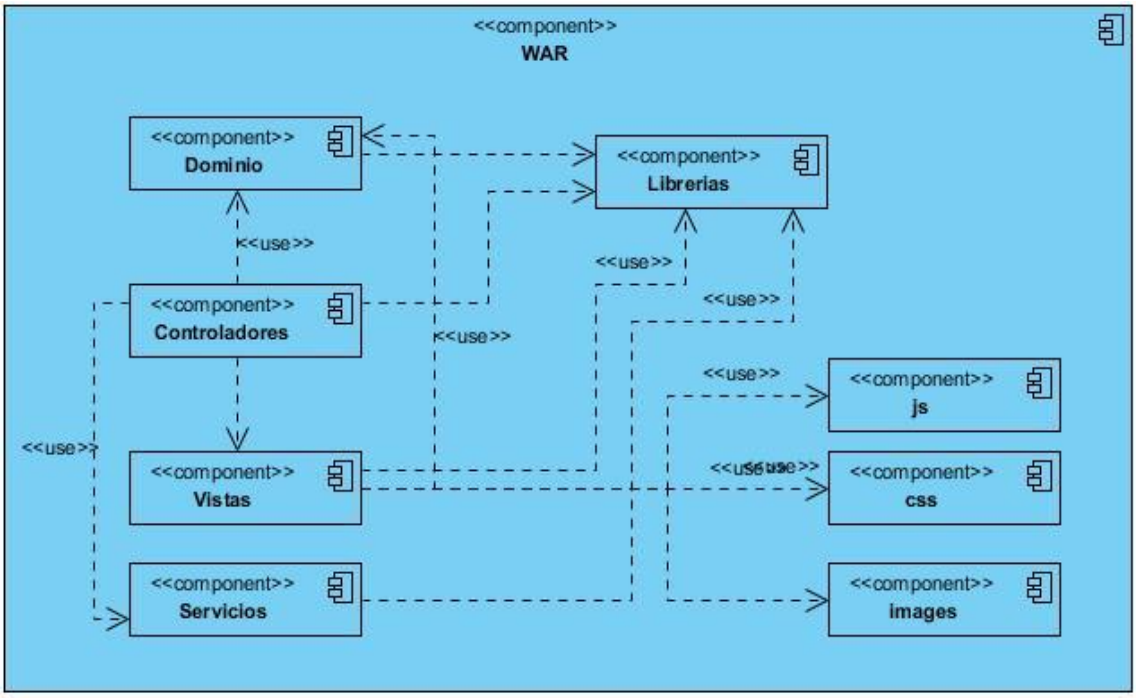


Figura 26. Diagrama de componentes del sistema

2.4.2.1 Descripción de los componentes

Componente Dominio

Propósito

Es el contenedor físico encargado de agrupar cada una de las clases que forman parte del dominio de la aplicación, archivos con extensión GROOVY.

Contenido

Categoría
MedidaTiempo
Motivo
MotivoSeleccionado
MsgEmergencia
MsgEmergenciaConfigurable
MsgEmergenciaPeriodico
MsgSitioConfiguracion
MsgSitioPublicacion
SitioPublicacion
Solicitud
Usuario
ValorTiempo

Componente Controladores

Propósito

Contenedor físico de las clases controladoras del sistema, archivos con extensión GROOVY.

Contenido

CategoriaController
MedidaTiempoController
MotivoController
MotivoSeleccionadoController
MsgEmergenciaConfigurableController
MsgmergenciaController
MsgEmergenciaPeriodicoController
SitioPublicacionController
SolicitudController
UsuarioController
ValorTiempoController

Componente Vistas

Propósito

Contenedor físico de las interfaces de usuario, archivos con extensión GSP.

Contenido

main.gsp
index.gsp
error.gsp
auth.gsp
denied.gsp

Y por cada clase de dominio este conjunto de vistas:
_form.gsp
create.gsp
list.gsp
edit.gsp
show.gsp

Componente Servicios

Propósito

Contenedor físico que agrupa cada uno de los servicios usados por los controladores, son archivos con extensión GROOVY

Contenido

AdministradorPublicacionMsgService
MsgEmergenciaService
SitioPublicacionService
ConfiguracionQuartzService
AdministradorSolicitudService

Componente js

Propósito

Contenedor físico de los ficheros que contienen código JavaScript.

Contenido

Application.js

Componente css

Propósito

Contenedor físico de los ficheros que contienen código CSS u hojas de estilos.

Contenido

errors.css

mobile.css

main.css

Componente images

Propósito

Contenedor físico de las imágenes de la aplicación.

Contenido

favicon.bpm

fondo.png

logo171.png

spinner.gif

2.5 Pruebas

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un sistema informático. (Pressman, 2001)

2.5.1 Métodos de prueba

Prueba de Caja Negra: Pruebas que se llevan a cabo sobre la interfaz del software, se centra principalmente en los requisitos funcionales del software. Pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto. Las pruebas de caja negra verifican las especificaciones funcionales y no consideran la estructura interna del programa, además

se hacen sin el conocimiento interno del producto y no se validan funciones ocultas. (Pressman, 2001)

2.5.2 Diseño de casos de prueba

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada y es a su vez la entidad más simple que siempre es ejecutada como una unidad, desde el comienzo hasta el final. Los casos de pruebas verifican si el producto satisface los requerimientos del usuario y si se comporta como se desea. Para comprobar el funcionamiento del sistema se realizaron diferentes casos de prueba de caja negra que a continuación se muestran.

Los diseños de casos de pruebas se muestran en el Anexo 2.

2.6 Conclusiones del capítulo

1. Se determinaron 20 requisitos funcionales y 7 no funcionales agrupados en casos de uso, que dieron paso al diseño del sistema.

2. Se seleccionó Grails como framework base de la aplicación que facilitó el desarrollo de aplicaciones web, reduciendo tiempo y esfuerzo por parte del equipo de desarrollo, logrando una mayor productividad utilizando implícitamente dos frameworks importantes en el desarrollo con Java: Spring e Hibernate y Grovvy como lenguaje de programación.

3. La arquitectura definida: Cliente – Servidor por ser una aplicación web, basada en la arquitectura en 3 capas propuesta por Grails, guió la posterior construcción del software.

4. Los casos de pruebas ejecutados arrojaron resultados satisfactorios, demostrando que el sistema desarrollado realiza todas las funcionalidades previstas.

Conclusiones

1. Después del análisis realizado a los sistemas destinados a la difusión de información de emergencia a nivel mundial, se determinó que los mismos no se ajustan a las necesidades de los Centros de Gestión a Emergencias 171.

2. El uso de Grails como framework base, facilitó el desarrollo de la aplicación, reduciendo tiempo y esfuerzo por parte del equipo de desarrollo, logrando una mayor productividad.

3. Se desarrolló un sistema que posibilita gestionar, dentro de los Centros de Gestión de Emergencias 171, la publicación de información de emergencia en Internet, contribuyendo a mantener a los ciudadanos actualizados en tiempo real.

4. Los casos de pruebas ejecutados arrojaron resultados satisfactorios, demostrando que el sistema desarrollado realiza todas las funcionalidades previstas.

Recomendaciones

Con los resultados obtenidos en la investigación y la experiencia adquirida en el desarrollo de la misma, se proponen las siguientes recomendaciones:

- Implementar las funcionalidades que permitan adicionar e instalar de forma dinámica nuevos plugins de interacción con otros sitios de publicación de información.
- Poner en práctica el sistema en todos los Centro de Gestión de Emergencia en Venezuela.

Bibliografía

Buschmann, Frank y otros. 1996. *Pattern-Oriented Software Architecture. John Wiley & Sons Inc.* Volume 1. 1996.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo del Software.* [trad.] Salvador Sánchez, y otros. Español. Madrid : Addison Wesley, 2000. ISBN: 84-7829-036-2.

Judd, Christopher, Nusairat Faisal, Joseph y James Shingler. 2008. *Beginning Groovy and Grails From Novice To Professional.* New York : s.n., 2008. s.n.

Letelier, Patricio y Penadés, Mª Carmen. Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP). [Citado el: 13 de Marzo de 2011.]. [En línea] Universidad Politécnica de Valencia. http://www.willydev.net/descargas/masyxp.pdf.

Nacho, Brito. 2009. Manual de desarrollo web con Grails. [Citado el: 20 de Abril de 2011.] [En línea] Junio de 2009. http://www.manual-de-grails.es. v1.0.4.

Pressman, Roger S. 2001. *Ingeniería del Software. Un enfoque Práctico.* [trad.] Darrel Ince. 5ta Edición. s.l. : Mc Graw Hill, 2001. pág. 614.

Referencias bibliográficas

911. 911Broadscat Emergency Notification Services. [En línea] [Citado el: 17 de Noviembre de 2011.] http://www.911broadcast.com/.

A. Malizia, cols. 2008. AlertOC Stay Informed. *SEMA4A: An ontology for emergency notification systems accessibility.* [En línea] 2008. [Citado el: 18 de Noviembre de 2011.] http://bos.ocgov.com/alertoc/alertoc.asp.

Academia, Real. Real Academia de la Lengua Española. [En línea] [Citado el: 23 de Noviembre de 2011.] http://www.rae.es/rae.html.

Apache Software Fundation. 1999-2012. The Apache Software Fundation. [En línea] [Citado el: 22 de Noviembre de 2011.] 1999-2012. http://tomcat.apache.org.

Buschmann, Frank y otros. 1996. *Pattern-Oriented Software Architecture .* 1996. Volume 1. [Citado el: 10 de Febrero de 2011.]

Cantillo, G, Palmera, R y Román, I. 2011. [En línea] 2011. [Citado el: 1 de Nomviembre de 2011.] http://www.actiweb.es/ipgcrp/pagina5.html.

Chalá, Bernal y Carolina, Nely. 2000. *Auditoría y reestauración de procesos seleccionados previo a la certificación ISO 9001:2000 en el Colegio Militar Eloy Alfaro Quito.* [En línea] 2000. [Citado el: 3 de Diciembre de 2012.]

Col. Computer Science Courseware. [En línea] [Citado el: 21 de Octubre de 2011.] http://www.gitam.edu/eresource/comp/gvr(os)/4.1.htm.

GITAM UNIVERSITY. GITAM UNIVERSITY. [En línea] http://www.gitam.edu/eresource/comp/gvr(os)/4.1.htm.

Ilustrados. [En línea] [Citado el: 29 de Noviembre de 2011.] http://www.ilustrados.com/publicaciones/EpZVVlyFyAbRDtMKhI.php.

Introducción a la Seguridad Ciudadana. **Col. 2011.** La Habana : Universidad de las Ciencias Informáticas, 2011.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo del Software.* [trad.] Salvador Sánchez, y otros. Español. Madrid : Addison Wesley, 2000. pág. 464. ISBN: 84-7829-036-2.

Judd, Christopher, Nusairat Faisal, Joseph y James Shingler. 2008. *Biginning Groovy and Grails From Novice To Professional.* New York : s.n., 2008. s.n.

Microsoft Corporation. 2011. Suscriptores de Oracle. [En línea] 2011. [Citado el: 19 de 01 de 2011.] http://msdn.microsoft.com/es-es/library/ms151738%28v=sql.100%29.aspx.

Nacho, Brito. 2009. Manual de desarrollo web con Grails. [En línea] Junio de 2009. http://www.manual-de-grails.es. v1.0.4.

National Emergency Alert Notification System. 1998-2005 Emcom National Emergency Alert Notification System. [En línea] [Citado el: 25 de Noviembre de 2011.] http://emcomus.org/index1.html.

New York University. 2011. CUNY Doctoral Program in Computer Science. [En línea] 2011. http://web.cs.gc.cuny.edu/~jniu/teaching/csc33200/files/0924-ProcessConceptAndState.pdf.

Nieto Alfonso. 2008. Difusión Informativa. [En línea] 2008. http://www.google.com.cu/url?sa=t&rct=j&q=concepto+de+difusion+de+informacion&source=web&cd=3&ved=0CFsQFjAC&url=https%3A%2F%2Fwww.unav.es%2Ffcom%2Fcomunicacionysociedad%2Fdescarga_doc.php%3Fart_id%3D303&ei=F5G6T6DiA4mCgwe4vPCnBQ&usg=AFQjCNFN11HkDUkkM7C3wI.

Orabel, S. 2004. "Manejadores de Base de Base de Datos - SQL, ORACLE, INFORMIX". [En línea] 2004. [Citado el: 28 de Noviembre de 2012.] http://www.ilustrados.com/publicaciones/EpZVVlyFyAbRDtMKhI.php.

Organización Nacional de Protección Civil. 13 de noviembre del 2001. *Ley de la Organización Nacional de Protección Civil y Administración de desastres.Decreto Presidencial N° 1.557.* República Bolivariana de Venezuela : Publicada en la Gaceta Oficial de la República Bolivariana de Venezuela, 13 de noviembre del 2001. 5.557.

Pasquali. 1979. Conceptos sobre Difusión, Divulgación, Periodismo y Comunicación. *Manuel Calvo Hernando.* [En línea] 1 de Noviembre de 1979. http://www.manuelcalvohernando.es/.

Pressman, Roger S. 2001. *Ingeniería del Software. Un enfoque Práctico.* [trad.] Darrel Ince. 5ta Edición. s.l. : Mc Graw Hill, 2001. pág. 614.

Rational Software Corporation. 2003. Glosario del Rational Unified Process. 2003. 2003.06.00.65.

Sanchez, María A. Mendoza. 2004. *Metodologías de desarrollo de software.* 2004.

Scribd. 2009. Scribd. [En línea] 2009. [Citado el: 2 de Noviembre de 2011.] http://es.scribd.com/doc/51996250/Definicion-de-Sistema-informatico.

slidershare. [En línea] [Citado el: 6 de Noviembre de 2011.] http://www.slideshare.net/lisluc18/definicin-de-sistema-informtico.

Smith, Glen y Ledbrook, Peter . 2009. *Grails in Action.* s.l. : Greenwich : Manning Publications, 2009.

Systems, Operating. 2001. Operating Systems. [En línea] 2001. [Citado el: 21 de noviembre de 2011.] http://stargazer.bridgeport.edu/sed/projects/cs503/Spring_2001/kode/os/process.htm.

TekAling. [En línea] [Citado el: 24 de Noviembre de 2011.] http://www.tekalign.com/.

Telemedik. 2003. Telemedik. [En línea] 2003. [Citado el: 9 de Noviembre de 2011.] http://www.telemedik.com/articulos.php?id=239.

Visual Paradigm International, Ltd. Visual Paradigm for UML. [En línea] [Citado el: 22 de 01 de 2011.]

—. 2010. *VPM-UML Quick Start.* s.l. : Visual Paradigm International., 2010.

Anexos

Anexo1. Expansión de los casos de uso

Nombre del CU	**Autenticar Usuario**
Actor	Usuario
Propósito	Verificar que el usuario es miembro del sistema.
Descripción	El caso de uso se inicia cuando el usuario decide acceder a la aplicación, por lo que debe autenticarse para garantizar la seguridad del sistema, inserta el usuario y la contraseña para acceder al sistema, se verifican los datos y finaliza cuando el sistema le otorga los privilegios según el rol que desempeña.
Precondiciones	El actor debe ser un usuario registrado en el sistema.
Postcondiciones	Se autenticó un usuario en el sistema.
Referencias	R1
Prioridad	Crítico
Sección "Autenticar usuario": Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
1. El usuario accede a la aplicación	2. Muestra la interfaz Autenticar usuario con los siguientes campos: ➢ Usuario ➢ Contraseña Y permite: ➢ Iniciar sesión
3. Introduce los datos requeridos.	
4. Selecciona la opción Iniciar sesión.	5. Valida los datos.
	6. Brinda privilegios de acceso.
	7. El caso de uso termina.
Flujos alternos	
5.a Existen campos vacíos	
	5. a.1 Envía el mensaje ("Debe llenar todos los campos"). 5. a.2 Regresa al paso 3 del Flujo Básico de eventos.
5.b Existen datos incorrectos	
	4. b.1 Envía mensaje ("El nombre de usuario o la contraseña son incorrectos.") 4. b.2 Regresa al paso 3 del Flujo Básico de eventos.

Nombre del CU	Cerrar sesión
Actor	Usuario
Propósito	Cerrar la sesión del usuario autenticado.
Descripción	El caso de uso comienza cuando el usuario decide cerrar su sesión. Realiza la operación y el caso de uso termina.
Precondiciones	Debe un usuario estar autenticado.
Postcondiciones	No aplica.
Referencias	R2
Prioridad	Crítico

Sección "Cerrar sesión": Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
1. Selecciona la opción Cerrar sesión.	2. Cierra la sesión del usuario.
	3. Regresa a la interfaz de inicio de sesión.
	4. El caso de uso termina.

Nombre del CU	Gestionar sitios de publicación
Actor	Administrador
Propósito	Crear, listar, habilitar y deshabilitar los sitios de publicación.
Descripción	El caso de uso comienza cuando el administrador decide adicionar un nuevo sitio de publicación al sistema, listar los existentes, habilitar o deshabilitar alguno de los registrados en el sistema. Realiza la operación deseada y el caso de uso termina.
Precondiciones	Para listar los sitios debe existir al menos uno en el sistema.
Postcondiciones	Se creó una nueva instancia en la base de datos del sitio de publicación creado.
Referencias	R3, R4, R5, R6
Prioridad	Crítico

Sección "Crear sitio": Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
1. Selecciona la opción Crear sitio.	2. Muestra la interfaz Crear sitio con los siguientes campos: ➢ Nombre del sitio. ➢ Url de acceso. ➢ Imagen de identificación ➢ Habilitado Y permite: ➢ Crear.

3. Introduce los datos del sitio y selecciona la opción Crear.	4. Valida los datos.
	5. Verifica si el sitio ya existe en el sistema.
	6. Registra el nuevo sitio.
	7. Muestra los datos del sitio creado.
	8. El caso de uso termina.
Flujos alternos	
4.a Existen campos vacíos	
	4. a.1 Envía el mensaje (“Debe llenar todos los campos”). 4. a.2 Regresa al paso 3 del Flujo Básico de eventos.
4.b Existen datos incorrectos	
	4. b.1 Envía mensaje (“Verifique los datos, hay datos incorrectos.”) 4. b.2 Regresa al paso 3 del Flujo Básico de eventos.
5. a El sitio ya existe dentro del sistema	
	5. a.1 Envía un mensaje (“El sitio ya existe dentro del sistema.”) 5. a.2 Regresa al paso 3 del Flujo Básico de eventos.
Sección “Habilitar/Deshabilitar sitios”: Flujo Básico de Eventos	
1. Selecciona la opción del menú Habilitar/Deshabilitar.	2. Lista los sitios registrados con sus datos correspondientes. Y permite: ➢ Habilitar/Deshabilitar
3. Habilita o deshabilita los sitios que desee.	4. Actualiza el criterio “habilitado” de los sitios modificados.
	5. El caso de uso termina.

Nombre del CU	**Gestionar mensaje**
Actor	Usuario
Propósito	Crear, editar, listar los mensajes de emergencia o mostrar su historial de publicación.
Descripción	El caso de uso comienza cuando el usuario decide elaborar un mensaje, editar la configuración de uno existente, listar los mensajes que han sido publicados en los sitios de difusión en Internet o mostrar su historial de publicación. Realiza la operación deseada y termina el caso de uso.
Precondiciones	Para listar los mensajes, debe haber sido publicado al menos uno.
Postcondiciones	Se creó una nueva instancia en la base de datos del mensaje de

	emergencia creado. Se modificó la instancia editada.
Referencias	R7, R8, R9, R10, R11, R12
Prioridad	Crítico

Sección "Elaborar mensaje": Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
1. Selecciona la opción Crear mensaje.	2. Muestra la interfaz Elaborar mensaje con los siguientes datos: ➢ Sitios de publicación ➢ Texto del mensaje Y permite: ➢ Configurable • Periódico ➢ Crear
3. Inserta los datos del mensaje.	
4. Selecciona la opción Crear	5. Valida los datos insertados por el usuario.
	6. Almacena el mensaje creado con los datos insertados.
	7. Invoca el "**CU Publicar mensaje**".
Flujos alternos	
5.a Existen campos vacíos	
	5. a.1 Envía el mensaje ("Debe llenar todos los campos"). 5. a.2 Regresa al paso 3 del Flujo Básico de eventos.
5.b Existen datos incorrectos	
	5. b.1 Envía mensaje ("Verifique los datos, hay datos incorrectos.") 5. b.2 Regresa al paso 3 del Flujo Básico de eventos.
2.b Opción "Configurable"	
	2.b.1 Muestra la interfaz Mensaje configurable con los siguientes datos: ➢ Fecha y hora de publicación.
2.c Opción "Periódico"	2.c.1 Muestra la interfaz Mensaje Periódico con los siguientes datos: ➢ Fecha y hora de publicación (Definido en la opción Configurable). ➢ Fecha y hora de expiración

	➢ Frecuencia de publicación (periodicidad)
Sección "Listar mensajes": Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
1. Selecciona la opción Listado de mensajes.	2. Muestra la interfaz Listar mensajes con todos los mensajes publicados y pendientes de publicación con los siguientes datos: ➢ Fecha y hora de creado ➢ Sitios donde fue publicado ➢ Texto del mensaje ➢ Usuario que lo registró Y permite: Para los mensajes con publicación configurada ➢ Editar mensaje ➢ Habilitar y deshabilitar mensaje ➢ Ver configuración de publicación Y para todos: ➢ Mostrar historial
	3. El caso de uso termina.
Flujos alternos	
2.a Opción "Editar mensaje"	**2.a.1** Invoca la sección **"Editar mensaje"**
2.b Opción "Mostrar historial"	**2.b.1** Invoca la sección **"Mostrar historial"**
2.c Opción "Habilitar/Deshabilitar"	**2.c.1** Invoca la sección **"Habilitar/Deshabilitar"**
2.d Opción "Ver configuración de publicación"	**2.d.1** Invoca la sección **"Ver configuración de publicación"**
Sección "Editar mensaje": Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
1. Selecciona la opción Editar mensaje.	2. Muestra la interfaz Editar mensaje con los siguientes datos: ➢ Sitios de publicación ➢ Fecha y hora de publicación ➢ Fecha y hora de expiración ➢ Periodicidad Y permite: ➢ Actualizar
3. Modifica los datos que considera necesarios.	
4. Selecciona la opción Actualizar	5. Valida los datos insertados.
	6. Actualiza los datos modificados.
	7. El caso de uso termina.
Flujos alternos	

5.a Existen campos vacíos	
	5. a.1 Envía el mensaje (“Debe llenar todos los campos”). 5. a.2 Regresa al paso 3 del Flujo Básico de eventos.
5.b Existen datos incorrectos	
	5. b.1 Envía mensaje (“Verifique los datos, hay datos incorrectos.”) 5. b.2 Regresa al paso 3 del Flujo Básico de eventos.
Sección “Mostrar historial”: Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
1. Selecciona la opción Mostrar historial.	2. Muestra el historial de publicación del mensaje con los siguientes datos: ➢ Sitio donde se publicó ➢ Fecha/Hora de publicación
	3. El caso de uso termina.
Sección “Habilitar/Deshabilitar”: Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
1. Selecciona la opción Habilitar/Deshabilitar en correspondencia del estado del mensaje.	2. Habilita o deshabilita el mensaje y actualiza el estado del mismo poniendo en funcionamiento o deteniendo su publicación.
	3. El caso de uso termina.
Sección “Ver configuración de publicación”: Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
1. Selecciona la opción Ver configuración.	2. Muestra la configuración de la publicación del mensaje con los siguientes datos: ➢ Fecha y hora de inicio de publicación ➢ Fecha y hora de fin de la publicación ➢ Periodicidad
	3. El caso de uso termina.

Nombre del CU	**Publicar mensaje**
Actor	Usuario
Propósito	Publicar el mensaje de emergencia en los sitios de publicación o almacenar la configuración de la publicación definida del mensaje creado.
Descripción	El caso de uso se inicia cuando el usuario crea el mensaje y es almacenado dentro del sistema. En el caso que dicho mensaje no esté creado para publicarse en al momento, se almacena la configuración de

	publicación del mismo para su posterior ejecución; sino, es enviado instantáneamente a los sitios de publicación definidos.
Precondiciones	Debe haberse elaborado previamente el mensaje de emergencia.
Postcondiciones	Queda almacenada en la base de datos la fecha y hora de publicación y el sitio al que fue publicado.
Referencia	R13
Prioridad	Crítico

Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
	1. Verifica que el mensaje se publica instantáneamente, o si tiene una configuración de publicación determinada.
	2. Si el mensaje no se publica instantáneamente, almacena los datos de la configuración de la publicación del mensaje para su posterior ejecución con los siguientes datos: ➢ Fecha y hora de inicio ➢ Fecha y hora de fin ➢ Periodicidad
	3. Verifica los sitios de publicación habilitados.
	4. Ejecuta los servicios de publicación por cada sitio de publicación habilitado.
	5. Almacena la publicación del mensaje con la fecha de publicación y el (los) sitio donde fue publicado.
	6. El caso de uso termina.
Flujos alternos	
1.a El mensaje tiene una fecha específica de publicación	
	1. a.1 Crea la configuración de publicación con la fecha establecida, y ejecuta la publicación cuando llega la fecha de publicación.
1.b El mensaje tiene un rango de fecha de publicación y una periodicidad determinada	
	1. b.1 Crea la configuración de publicación con la fecha de inicio, fecha de expiración y la periodicidad establecida. Ejecuta la publicación cuando llegue la fecha de inicio, cada vez que se cumpla la periodicidad hasta que llegue la fecha de expiración.

Nombre del CU	**Mostrar solicitudes de emergencia**

Actor	Usuario
Propósito	Mostrar un listado de las solicitudes registradas en el centro, dados los motivos seleccionados.
Descripción	El caso de uso comienza cuando el usuario desea ver las solicitudes de emergencia de los motivos seleccionados anteriormente, registradas en el SIGESC que están dentro del rango de consulta definido por el administrador o crear un mensaje a partir de las mismas. El usuario realiza la operación deseada y el caso de uso termina.
Precondiciones	Deben haberse seleccionado las categorías de motivos de los cuales se mostrarán las solicitudes registradas asociadas a las mismas.
Postcondiciones	No aplica.
Referencias	R17, R18
Prioridad	Crítico

Sección "Mostrar solicitudes de emergencia": Flujo Básico de Eventos	
Acción del Actor	**Respuesta del sistema**
1. Selecciona la opción Mostrar solicitudes	2. Realiza una búsqueda de las solicitudes registradas con los siguientes criterios de búsqueda: ➢ En el rango definido por el administrador comenzando a partir de la hora de petición de búsqueda. ➢ Dentro de las categorías y motivos seleccionados.
	3. Muestra un listado de las solicitudes que coinciden con la búsqueda realizada con los siguientes datos: ➢ Dirección ➢ Descripción de la solicitud Y permite: ➢ Elaborar mensaje ➢ Ver detalles
	4. El caso de uso termina.
Flujos alternos	
3.a Elaborar mensaje	
	3. a.1 Invoca el "**CU Gestionar mensaje**" en la sección Elaborar mensaje.
3.b Ver detalles	

	3.b.1 Muestra los detalles de los datos de la solicitud seleccionada con los siguientes datos: ➢ Categoría y motivo al que está asociado ➢ Fecha de registro ➢ Dirección ➢ Descripción de la solicitud

Nombre del CU	**Seleccionar motivos de solicitud**
Actor	Administrador
Propósito	Seleccionar los motivos de solicitud de emergencia de los que se desee ver las solicitudes registradas en el centro.
Descripción	El caso de uso comienza cuando el administrador decide seleccionar los motivos de solicitud asociados a una categoría para luego poder ver las nuevas solicitudes de emergencia registradas en el centro asociadas a dicho motivo. Realiza la operación y el caso de uso termina.
Precondiciones	No aplica.
Postcondiciones	Se almacenan dentro de la base de datos los motivos de solicitud seleccionados.
Referencias	R14, R15, R16
Prioridad	Crítico

Sección "Seleccionar motivos": Flujo Básico de Eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción Seleccionar motivos	2. Muestra la interfaz Configurar motivos con el listado de categorías y los motivos asociados a las mismas.
3. Selecciona los motivos de los cuales quiere ver las solicitudes registradas.	4. Almacena los motivos seleccionados para su posterior uso.
	5. El caso de uso termina.

Nombre del CU	**Listar categoría y motivos de solicitud**
Actor	Usuario
Propósito	Listar las categorías y los motivos asociados registrados en la base de datos del SIGESC.
Descripción	El caso de uso comienza el usuario se autentica y es iniciada su sesión, el sistema busca en la base de datos las categorías y los motivos y las solicitudes de emergencia asociados. Muestra las categorías de motivo con la cantidad de solicitudes asociadas a las mismas y el caso de uso termina.
Precondiciones	Debe estar autenticado el usuario.

Postcondiciones	No aplica.
Referencias	R13, R14
Prioridad	Crítico

Sección "Listar categorías y motivos": Flujo Básico de Eventos	
Acción del actor	**Respuesta del sistema**
1. Inicia su sesión autenticándose en el sistema.	2. Carga las categorías registradas en la base de datos y los motivos asociados a las mismas.
	3. Muestra el listado de las categorías de motivos con la cantidad de solicitudes correspondientes.
	4. El caso de uso termina.

Nombre del CU	**Gestionar intervalos de tiempo**
Actor	Administrador
Propósito	Configurar los intervalos de tiempo que corresponden a la periodicidad de la publicación de los mensajes periódicos.
Descripción	El caso de uso comienza cuando el administrador se dispone a modificar los intervalos de valores de tiempo definidos en el sistema. Realiza la operación y el caso de uso termina.
Precondiciones	No aplica.
Postcondiciones	Se modifican en la base de datos los intervalos de tiempo que se modifiquen.
Referencias	R19
Prioridad	Crítico

Sección "Editar intervalo de tiempo": Flujo Básico de Eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción Intervalos de periodicidad.	2. Muestra la interfaz Editar intervalos de tiempo con los siguientes datos: ➢ Medida (Horas, Minutos, Segundos) ➢ Mínimo ➢ Máximo Y permite: ➢ Actualizar.
3. Introduce los datos correspondientes.	
4. Selecciona la opción Actualizar.	5. Valida los datos.
	6. Actualiza los datos del nuevo intervalo de tiempo.

	7. El caso de uso termina.
Flujos alternos	
5.a Existen campos vacíos	
	5. a.1 Envía el mensaje (“Debe llenar todos los campos”). 5. a.2 Regresa al paso 3 del Flujo Básico de eventos.
5.b El valor de tiempo es menor que cero	
	5. b.1 Envía el mensaje (“El intervalo de tiempo no puede ser cero ni negativo”).
Sección “Listar valores de tiempo”: Flujo Básico de Eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción Listar intervalos de tiempo	2. Muestra los intervalos de tiempo asociados a horas, minutos y segundos.
	3. El caso de uso termina.

Nombre del CU	**Configurar tiempo de consulta**
Actor	Administrador
Propósito	Configurar el período de búsqueda de las solicitudes de emergencia registradas en el SIGESC.
Descripción	El caso de uso comienza cuando el administrador se dispone a editar el intervalo de búsqueda de las solicitudes determinado en el sistema. Realiza la operación y el caso de uso termina.
Precondiciones	No aplica.
Postcondiciones	Se modifican en la base de datos el período de búsqueda de las solicitudes de emergencia.
Referencias	R20
Prioridad	Crítico
Sección “Configurar tiempo de consulta”: Flujo Básico de Eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción Intervalo de búsqueda.	2. Muestra la interfaz Editar rango con los siguientes datos: ➢ Rango Y permite: ➢ Actualizar
3. Modifica el valor del rango de búsqueda definido.	
4. Selecciona la opción Actualizar.	5. Valida los datos.
	6. Actualiza los datos del nuevo rango de

	búsqueda.
	7. El caso de uso termina.
Flujos alternos	
5.a Existen campos vacíos	
	5. a.1 Envía el mensaje (“Debe llenar todos los campos”). 5. a.2 Regresa al paso 3 del Flujo Básico de eventos.
5.b El rango de búsqueda es menor que cero	
	5. b.1 Envía el mensaje (“El rango de búsqueda no puede ser cero ni negativo”).