1493/XXX-2016

San Juan de Pasto, 15 de diciembre de 2016

Arquitecto

**DARIO ANDRES GOMEZ CABRERA**

Dirección Para la Gestión del Riesgo de Desastres

Referencia: Respuesta propuesta APP Gestión del Riesgo

Cordial saludo,

Teniendo en cuenta que la Subsecretaría de Sistemas de Información, es la encargada de brindar soporte tecnológico para el normal desarrollo de los procesos informáticos de las diferentes dependencias de la Administración Municipal y que además ha establecido lineamientos para el desarrollo y mantenimiento de aplicativos, Me permito informar que después de analizar la propuesta enviada por la empresa “*IDEAR TICS”* debemos recomendar lo siguiente:

1. En la propuesta el punto *3.1 Módulo de administración web*, no establece la arquitectura que se usará para su implementación y de acuerdo a los lineamientos que se ha establecido desde esta oficina todo desarrollo que se implemente en los servidores de la alcaldía tiene que ser construido en una arquitectura MVC que permita su fácil entendimiento y mantenimiento, para mayor información anexamos al final el documento de **arquitectura de desarrollo de aplicaciones web**, el cual contempla el uso de symfony y angular para el desarrollo de aplicativos para la entidad.
2. El desarrollo de una APP nativa evita que usuarios que usan otras plataformas accedan al servicio, por lo cual se recomienda que se piense en una APP hibrida o una aplicación con responsive design.
3. La licencia tiene que establecer que la alcaldía de Pasto puede modificar el código para posteriores arreglos, mejoras o mantenimiento y de esta forma permitir que la herramienta siga prestando sus servicios a los ciudadanos, sin tener que depender de terceros.
4. Establecer cuales son los requisitos que tienen la Dirección Para la Gestión del Riesgo de Desastres, y así establecer un desarrollo que se ajuste a las necesidades de esta oficina y a los requerimientos que tienen los ciudadanos en cuanto a información, anexamos información de APPS que se utilizan en gestión del riesgo a nivel mundial para tener un referente de las necesidades que deseamos cubrir.

**Jakarta Flood Alert** (de Indonesia): Esta aplicación móvil controla los niveles actuales de agua, el cambio en las últimas seis horas y otros indicadores de 14 compuertas. Los usuarios pueden obtener la información más reciente sobre el estado de las esclusas y las probabilidades de futuras inundaciones en lugares específicos, y compartirla a través de las redes sociales para informar y preparar a otras personas en la zona.

https://www.youtube.com/watch?v=BI5wFjMIY0I

**Nigechizu** (de Japón): Esta herramienta web, cuyo nombre significa “mapa de evacuación”, digitaliza los mapas creados de manera colaborativa en talleres comunitarios para indicar los tiempos de evacuación. La versión digital, que es interactiva, permite evaluar los tiempos de salida para las rutas determinadas por los usuarios, proponer rumbos alternativos y apoyar a los planificadores urbanos y las autoridades encargadas de la gestión de desastres.

https://vimeo.com/96909505

**Quick Disaster** (de Indonesia): Esta aplicación creada para dispositivos portátiles contiene una variedad de funciones para entregar a los usuarios información sobre desastres. Por ejemplo, mediante el uso de datos de localización en tiempo real, quienes la usan reciben una advertencia cuando entran en zonas propensas a cierto tipo de catástrofes, y se les muestran rutas de evacuación cuando ocurre un desastre.

https://www.youtube.com/watch?v=zbWK7lWEP7A

La información se puede encontrar en la página:

http://www.bancomundial.org/es/news/feature/2014/06/30/innovative-apps-for-disaster-risk-reduction-win-global-attention

Atentamente,

**JHONNATAN RODRIGO HUERTAS**

Subsecretario de Sistemas de Información

**ARQUITECTURA DESARROLLO DE APLICATIVOS WEB**

**VISTA LÓGICA**

Se deberá contar con cuatro componentes disponibles que le permitirán realizar la recepción, validación, procesamiento y difusión de los datos de una manera ágil y sencilla.

1. Componente de gestión de datos

* Organización de los datos: en función de los requerimientos de información para los análisis (estructuras de datos)
* Cargue de los datos: ingreso de archivos planos validados en la estructura de almacenamiento definida.
* Administración de fuentes de datos: crea y mantiene actualizada una base de datos sobre las fuentes de datos, con los insumos que provee el ejercicio de identificación y evaluación de fuentes, y los procedimientos establecidos para la entrega, recepción y validación. La base de datos debe poder categorizar las fuentes en: sistemáticas y no sistemáticas.
* La actualización incluye: ingresar nuevas fuentes, desactivar fuentes actuales, desactivar datos de una fuente, modificar periodicidad, modificar forma o procedimiento de transferencia, modificar validaciones.
* Recepción y validación de datos: incluye los procesos manuales o automáticos (web services) para recibir los datos provenientes de distintas fuentes, y llevar las estadísticas de: número de fuentes de las cuales se ha recibido, número de registros recibidos, número de registros útiles (que pasaron las validaciones) fechas de recepción, etc.

1. Componente de gestión de información:

* Definición de formatos para registrar los requerimientos de información, en términos de: cruces de variables, presentación de las salidas, procedimientos estadísticos que se deben realizar sobre los datos.
* Organización de la recepción de requerimientos: registro de los requerimientos y de las fechas de entregas parciales y finales.
* Procesamiento de los datos según los requerimientos, realización de pruebas de confiabilidad, y organización de los resultados según la forma de presentación convenida.

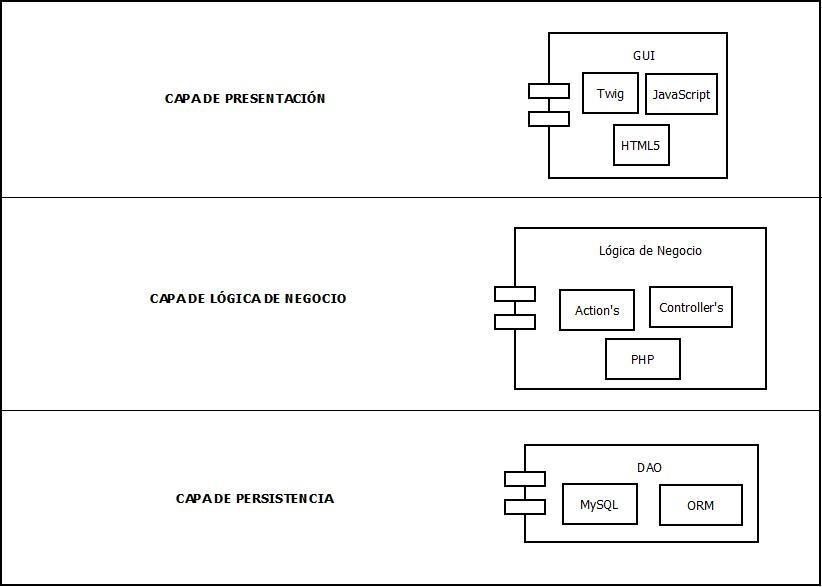
1. Componente de análisis y difusión de información:

* Realización del análisis integral
* Disposición del portal de información
* Definición y puesta en funcionamiento de los procedimientos para la administración la plataforma para actualización de la información.

1. Componente de soporte informático:

* Administración de Bases de datos
* Administración de servicios: hosting, internet.
* Administración de Seguridades.

Modelo de arquitectura MVC, que reconoce 3 capas; capa de presentación, capa de negocio y capa de persistencia, cada una con funcionalidades propias y diferenciadas pero no excluyentes que permiten manejar de forma más modular el desarrollo de la aplicación, por ello se dice que se basa en el uso de componentes ampliamente modulares.



**VISTA DE IMPLEMENTACIÓN**

Descripción: La vista está comprendida en módulos, cada uno de estos módulos contiene por lo menos 7 tipos de clases (Gestores DAO, Entidades DAO, Gestor BL, Acción Entidad, Acción Presenta Entidad, Entidad, Presentacion Entidad).

Cada módulo se encarga de un grupo específico de funcionalidades del sistema. Y a continuación se explica su funcionamiento

* Gestores DAO: Esta clase se encarga de interactuar con la base de datos y no contiene lógica de negocio, pero si lógica de persistencia.
* Entidades DAO: Esta clase contiene los comandos SQL y se encarga de interactuar directamente con la base de datos.
* Entidad: Es la representación lógica de un concepto del negocio.
* Gestor BL: Se encarga de la lógica del negocio, realiza las validaciones respectivas e invoca al gestor de base de datos.
* Acción Gestor Entidad: Contiene las acciones que se pueden realizar en la capa de presentación, se encarga de recibir la petición y llama GestorBL para que realice la lógica de negocio correspondiente. Esta clase interactúa directamente con la capa de presentación.
* Acción Presentación Entidad: Esta clase devuelve la información solicitada por el usuario haciendo uso de los métodos que contiene la clase GestorBL. Se comunica directamente con la capa de presentación.
* Presentación Entidad: Es aquella que contiene los datos que visualizará el usuario.

**CAPA DE PRESENTACIÓN**

Esta capa se encarga de la presentación de los datos lo cuales fueron obtenidos a través la lógica del negocio, la clase que pertenece a esta capa es la Presentación Entidad.

**Twig**

El usuario interactúa con las interfaces de la aplicación a través de los formularios y las validaciones. Twig compila las plantillas a PHP optimizando código. La sobrecarga en comparación con código PHP regulares se reduce al mínimo. Twig es una tecnología para crear aplicaciones Web.

**JavaScript**

La tecnología de JavaScript es un lenguaje de programación interpretado, Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargando junto con el códigoHTML.

**CAPA DE NEGOCIO**

**Identificación de Clases**

Esta capa se encarga de la lógica del negocio, es el intermediario entre la capa de base de datos y la capa de presentación, las clases que pertenecen a esta capa son: GestorBL, AccionGestor, AccionPresenta.

**Symfony 2**

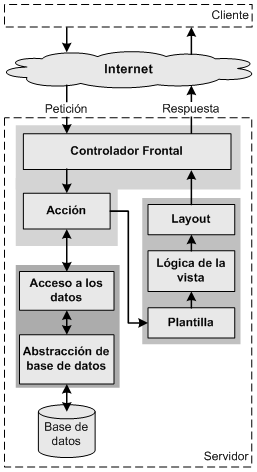
Es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web.

Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente en PHP 5.3. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel.

Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas (Unix, Linux, etc.) como en plataformas Windows.

Las características más comunes para el desarrollo de proyectos web están automatizadas en symfony, tales como:

* Permite la internacionalización para la traducción del texto de la interfaz, los datos y el contenido de localización.
* La presentación usa templates y layouts que pueden ser construidos por diseñadores de HTML que no posean conocimientos del framework.
* Los formularios soportan la validación automática, lo cual asegura mejor calidad de los datos en las base de datos y una mejor experiencia para el usuario.
* El manejo de cache reduce el uso de banda ancha y la carga del servidor.
* La facilidad de soportar autenticación y credenciales facilita la creación de áreas restringidas y manejo de seguridad de los usuarios.
* El enrutamiento y las URLs inteligentes hacen amigable las direcciones de las páginas de la aplicación.
* Las listas son más amigables, ya que permite la paginación, clasificación y filtraje automáticos.
* Los plugins proveen un alto nivel de extensibilidad.
* La interacción con JavaScript es mucho más sencilla.
* Rápido y consumo menor de memoria: Un error común en el desarrollo de aplicaciones es la de considerar el rendimiento o velocidad de la aplicación cuando la implementación ha terminado. Symfony2 fue concebido desde el principio para favorecer el rendimiento. Symfony2 es aproximadamente tres(3) veces más rápido que la versión 1.4 o de 1.10 de Zend Framework, usando la mitad de consumo de memoria.
* Flexibilidad ilimitada: Symfony2 es adaptable.
* Gestión de la complejidad: Permite desarrollar aplicaciones complejas con múltiples funcionalidades.
* Pieza a pieza: Permite construir de acuerdo con las funciones requeridas.
* Soporte: Proporcionado por Sensio, adicionalmente la comunidad (listas de correo, IRC, etc) y las empresas de servicios que han invertido en este framework.
* Por último, también es con miras a un desarrollo sostenible que Symfony se distribuye bajo licencia Open Source MIT, que no impone restricciones y permite el desarrollo de código abierto, así como aplicaciones propietarias.
* Ampliable: Desde la más pequeña pieza a la base completa en sí, todo lo que se presenta como un "paquete" (o plug-in en el lenguaje Symfony) en Symfony2. Cada paquete está destinado para añadir funcionalidad al framework, por supuesto, y cada paquete también puede ser reutilizada en otro proyecto o compartida con el resto de la comunidad. En cualquier caso, el sistema permite todo cambio dentro de Symfony, incluyendo el propio núcleo. Uso de contratos de la interfaz del sistema entre las piezas, el comportamiento del framework así puede ser cambiado a voluntad, sin necesidad de reconfiguración completa.
* Estable y sostenible: Desarrollado por los Laboratorios Sensio, las principales versiones de Symfony son soportados por 3 años por la empresa. E incluso de por vida en cuanto a las cuestiones relacionadas con la seguridad se refiere. Para mayor estabilidad, las versiones menores del contrato Symfony2 y la interfaz también están garantizadas y la compatibilidad entre todas las versiones secundarias se llevará a cabo en el API definido por las interfaces públicas.

**CAPA DE PERSISTENCIA**

**Identificación de Clases**

Esta capa se encarga del acceso de datos, interactúa con la base de datos, las clases que pertenecen a esta capa son el Gestor DAO y la Entidad DAO.

**Doctrine ORM**

Doctrine es un mapeador de objetos-relacional (ORM) escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una capa de abstracción que se sitúa justo encima de un SGBD (sistema de gestión de bases de datos).

Una característica de Doctrine es el bajo nivel de configuración que necesita para empezar un proyecto. Doctrine puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas. No es necesario generar o mantener complejos esquemas XML de base de datos como en otros frameworks.

Otra característica importante de Doctrine es la posibilidad de escribir consultas de base de datos utilizando un dialecto de SQL denominado DQL (Doctrine Query Language) que está inspirado en Hibernate (Java).

Otras características notables:

* Soporte para datos jerárquicos.
* Soporte para hooks (métodos que pueden validar o modificar las escrituras y lecturas de la base de datos) y eventos para manejar la lógica de negocio relacionada.
* Herencia.
* Un framework de caché que utiliza diversos motores como memcached, SQLite o APC.
* Transacciones ACID.
* Diversos comportamientos del modelo (conjuntos anidados, internacionalización, log, índice de búsqueda).
* Una función "compilar" que combina varios archivos PHP del framework en uno solo para evitar el descenso de rendimiento que provoca incluir varios archivos PHP.

**Vista Despliegue**

La calidad del servicio brindado al usuario será el mismo aun cuando aumente la cantidad de transacciones en el sistema. La solución es lo suficientemente extensible, la separación del contenido web dinámico del estático garantiza ello, si se desea agregar una nueva funcionalidad o una integración a otro sistema bastaría con agregar un nodo especializado (web service) para ello.

**Tamaño y Rendimiento**

La aplicación no demanda una gran cantidad de espacio, se tiene en cuenta la cantidad de usuarios que van a hacer uso de ella. En cuanto al rendimiento del sistema, los tiempos de carga y descarga de elementos se mantiene lo más bajo posible para incentivar el uso de la página.

**Calidad**

La arquitectura del software se ha diseñado para ser independiente de la plataforma en la que el sistema se utilice, ya que esto ofrece al sistema la capacidad de ser portable.

La aplicación está desarrollada por capas, de manera tal que las más superficiales, como la de interfaz, no afectarán a la capa lógica. Cada capa se comunicará con las capas que estén directamente por debajo de ellas. No existen saltos de acceso entre capas de manera tal que se mantenga una eficiente comunicación entre ellas.

**Manejo de Errores**

En el desarrollo del software fue importante crear una buena política de manejo de errores de tal manera que se cubren la mayoría de los posibles casos de mal funcionamiento durante el uso de todas las funciones que ofrece el sistema. Además, la información que puede ofrecer una buena política de manejo de excepciones es muy valiosa a la hora de depurar el sistema y por lo tanto, el mantenimiento puede simplificarse a gran escala.

**Buenas Prácticas**

Para el manejo de errores se seguirán las siguientes prácticas:

* Encapsulamiento: en el caso de que una excepción viaje a través de las capas del sistema y se cree alguna otra a partir de la primera generada, se guarda toda la información del primer error a través del encapsulamiento en el momento de la creación de la nueva excepción. De esta forma, no se pierde información debido a la propagación de la excepción o de la sustitución a través de las capas del sistema.
* Lanzar temprano: las excepciones son lanzadas en el preciso instante en el que ocurre el error. Así se tiene un mejor control a la hora de depurar y de buscar el origen de la falla.
* Atrapar tarde: las excepciones se atraparán cuando se tenga la mayor cantidad de información posible y cuando se esté en la capa correspondiente del sistema que tenga las herramientas necesarias para manejar correctamente el error.
* Obtener y desplegar la mayor cantidad de información: a la hora de generar un error, la mayor cantidad de información debe ser desplegada de tal manera que se pueda ubicar fácilmente el origen y por lo tanto, se depure rápidamente el sistema

**POLÍTICA DEL MANEJO DE EXCEPCIONES**

**Capa de Persistencia**

En esta capa se utilizará una política de Propagación y de encapsulado de excepción. Esto tendrá como consecuencia que la información del error ocurrido no se pierda y se transmita lo más posible a través de su viaje por las capas del sistema.

**Capa Lógica**

Para la capa lógica se aplicará una política de sustitución y encapsulamiento. Se sustituye la excepción proveniente de la capa inferior de tal manera que se cree alguna otra más específica a la causa del error y que, por consiguiente, brinde información más específica a la capa superior.

**Capa de Presentación**

En la capa de interfaz se aplicará la política de manejo de errores. En esta capa del sistema se realizará el manejo de todas las excepciones provenientes de las capas inferiores y se desplegará al usuario la información necesaria al error ocurrido.

Seguridad

El sistema de seguridad del software se basará en identificar primero al usuario (autenticación) y comprobando después si ese usuario tiene acceso al recurso solicitado (autorización).

**Firewalls (autenticación)**

El sistema de seguridad del sistema se activa cuando un usuario hace una petición a una URL que está protegida por un firewall o cortafuegos. El trabajo del firewall determina si el usuario necesita estar autenticado, y si lo necesita, enviar una respuesta al usuario para iniciar el proceso de autenticación.

El firewall se activa cuando la URL de una petición entrante concuerda con el valor de su opción de configuración. No obstante, el hecho de que el firewall esté activado no significa que el navegador muestra la caja de usuario y contraseña para todas las URL. Los usuarios pueden acceder sin que la aplicación les pida que se autentiquen.

Este funcionamiento es posible en primer lugar porque el firewall permite el acceso a los usuarios anónimos debido a la opción de configuración anonymous. En otras palabras, el firewall no exige que todos los usuarios se autentiquen para acceder a algunos espacios de la aplicación.

**Control de acceso (autorización)**

Los roles son la clave del sistema de autorización: el usuario puede acceder a una URL sólo si cuenta con el rol autorizado.

Al igual que sucedía anteriormente, cuando el usuario realiza su petición, el firewall no solicita ningún tipo de identificación. Sin embargo, en cuanto la capa de control de acceso deniega el acceso al usuario (porque los usuarios anónimos no cuentan con el rol solicitado), el firewall toma el control de la aplicación e inicia el proceso de autenticación.

Se utiliza el método de autenticación con formulario, el usuario será redirigido a la página de inicio de sesión.

Ahora el usuario tiene la oportunidad de enviar sus credenciales a la aplicación. Si estas credenciales son válidas, se reintenta la petición original.

**Captura de datos**

Depuración oportuna de los datos por integración a la estructura de entidades directamente.

**Usabilidad**

El sistema se basa en tres (3) reglas adaptadas en la plataforma web, que permitan considerar el software como un sistema web "usable".

* Rápido - Las páginas se cargan en una media de 4 segundos. Los usuarios lo más que esperarán en ver el contenido de una página web es de una media de 10 segundos.
* Simple – La aplicación mantiene una navegación constante. No fuerza a los visitantes a aprender diversos caminos o esquemas para la navegación en diversas partes de su sitio. No abusa de la utilización de la animación y utiliza componentes CSS3.
* Para la mayoría – El sistema Web es compatible con todos los navegadores y ordenadores para su fácil usabilidad.