Sistema integrado de información

Version 1.3

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 07/09/14 | 1.0 | Construcción del documento | Daniel, Dival |
| 08/09/14 | 1.1 | Inclusión vistas de caso de uso y lógica | Daniel, Dival |
| 09/09/14 | 1.2 | Inclusión vistas de despliegue e implementación. | Daniel, Dival |
| 09/09/14 | 1.3 | Inclusión de las características de calidad. | Daniel, Dival |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Architectural Representation 4

3. Architectural Goals and Constraints 4

3.1 Plataforma técnica 4

3.2 Seguridad 4

3.3 Persistencia 5

3.4 Disponibilidad 5

4. Use-Case View 5

5. Logical View 5

5.1 Overview 5

5.2 Architecturally Significant Design Packages 6

5.3 Use-Case Realizations 6

6. Deployment View 6

7. Implementation View 7

*7.1* Vista 7

*7.2* Servidor web 7

7.3 Servidor BD 7

7.4 Diagrama de componentes 8

8. Size and Performance 8

9. Quality 8

# Introduction

## Purpose

El presente documento tiene como propósito mostrar una visión comprensible de la arquitectura global del software haciendo uso de diferentes vistas arquitectónicas para definir diferentes aspectos del sistema.

## Scope

El alcance de este documento es el de definir la arquitectura que se utilizara en el desarrollo del proyecto, mostrándolas a través de diferentes diagramas y vistas que la metodología RUP facilita para este propósito.

## Definitions, Acronyms, and Abbreviations

RUP: Rational Unified Process.

## References

IBM Rational Unified Process (2014) <http://www-01.ibm.com/software/rational/rup/>, obtenido el 13 de agosto de 2014.

Bass Len, Clements Paul, Kazman Rick (2003). Software Architecture in practice, Second edition.

## Overview

En el documento se define la arquitectura a utilizar en el desarrollo del proyecto basándose en el modelo “4+1” definiendo los diferentes diagramas que este requiere y los parámetros de calidad para el correcto desarrollo de la aplicación.

# Architectural Representation

En este documento se detalla la arquitectura utilizando las vistas que se definen en el modelo “4+1” de Kruchten, el cual contiene las vistas lógica, de desarrollo, física, de escenarios y de casos de uso, utilizando las nomenclaturas propuestas por RUP.

**Vista lógica:** Apoya los requisitos funcionales (lo que el sistema debe brindar en términos de servicios a los usuarios) con el modelo de objetos de diseño y la realización de los casos de uso.

**Vista de desarrollo:** Se centra en la organización de los módulos en el ambiente de software, se organizan en jerarquía de capas.

**Vista física:** Toma en cuenta los requisitos no funcionales del sistema como lo son la disponibilidad, confiabilidad, performance y escalabilidad.

**Escenarios:** El acoplamiento de las diferentes vistas.

**Vista de casos de uso:** Muestra los casos de uso que representan las funcionalidades principales para el funcionamiento de la aplicación.

# Architectural Goals and Constraints

En esta sección se describen los requerimientos y objetivos que poseen algún tipo de impacto en la arquitectura.

## Plataforma técnica

El desarrollo de la aplicación se realizara en el lenguaje de programación PHP utilizando el framework de desarrollo Yii, el cual provee casi todas las funcionalidades que se necesitan en aplicaciones web 2.0 y es un framework MVC.

## Seguridad

Un factor supremamente importante en el sistema debido a la naturaleza de la organización es el manejo de usuarios, los cuales podrán tener diferentes roles, por esto es crucial un control de usuarios que se implementara con un módulo de autenticación y registro en la comunidad. Para esto se debe manejar tanto seguridad en el ingreso como de los datos.

## Persistencia

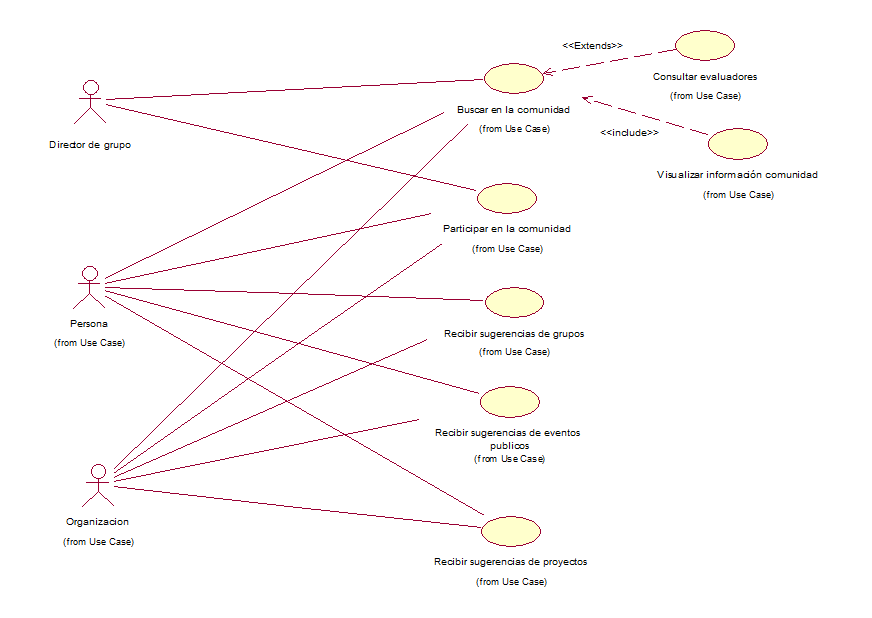
La persistencia de datos será asegurada usando el sistema gestor de bases de datos mySQL. El cual es relacional, multi-hilo y multiusuario, además es compatible con el framework Yii.

## Disponibilidad

La disponibilidad del sistema es un requisito clave por la naturaleza de la organización ya que es un sistema que opera a nivel nacional con una comunidad muy grande, la arquitectura debe garantizar la disponibilidad del servicio sobre todo en los momentos en los que se utilizan las funcionalidades core de la aplicación.

# Use-Case View

En esta sección se listara el modelo de casos de uso que representan funciones importantes en la implementación final de la aplicación. El caso de uso con mayor impacto es el de participar en la comunidad ya que este es el que permite el registro de los diferentes usuarios de la aplicación.



# Logical View

## Overview

Para el desarrollo de la aplicación se realizará en tres capas las cuales son el modelo, la vista y el controlador siguiendo el patrón MVC.

## Architecturally Significant Design Packages

El modelo arquitectónico “modelo vista controlador”, es un diseño de aplicaciones compuesto por:

Modelo: Contiene la funcionalidad y el estado de la aplicación.

Vista: Es la presentación del modelo que interactúa con el usuario.

Controlador: Reacciona a la petición del cliente utilizando el método adecuado para la función necesitada.



Modelo MVC tomado de: <http://librosweb.es/jobeet_1_4/capitulo_4/la_arquitectura_mvc.html>

## Use-Case Realizations

Ver artefacto UseBuscarEnLaComunidad.

Ver artefacto UseConsultarEvaluadores.

Ver artefacto UseParticiparEnLaComunidad.

Ver artefacto UseRecibirSugerenciasEventosPublicos.

Ver artefacto UseRecibirSugerenciasGrupos.

Ver artefacto UseRecibirSugerenciasProyectos.

Ver artefacto UseVisualizarInformacion.

# Deployment View

En el diagrama de despliegue se muestra el ambiente en el que el aplicativo quedara finalmente. Siguiendo el patrón MVC vemos que los clientes necesitan una maquina simple conectada a internet con un navegador para acceder a la vista, cuando el usuario interactúa con la vista un servidor web capta sus instrucciones y las ejecuta, este, que hace el rol de controlador accede a los datos que están alojados en otro servidor, una vez obtenidos los datos envía respuesta al usuario. En esta ambiente es crucial la estructura de la red entre los servidores y la disponibilidad con los usuarios.



# Implementation View

## Vista

En esta capa se encuentran las diferentes interfaces desarrolladas específicamente para la interacción con el usuario ya sea recibiendo o mostrando los datos.

## Servidor web

En esta capa se aloja toda la lógica del proyecto, en esta se encuentran los controladores que proveen todas las funcionalidades necesarias por la aplicación que se pueden juntar en tres grandes grupos los cuales son la búsqueda, la visualización de información y el registro como un nuevo usuario de la aplicación, las funcionalidades antes nombradas son administradas por las herramientas proporcionadas por el framework Yii.

## Servidor BD

En esta capa se aloja el motor de base de datos el cual alojara la información referente a la aplicación y lo datos recuperados de las dos bases de datos que se utilizaban antiguamente.

## Diagrama de componentes

|

# Size and Performance

* Ofrecer un soporte 24/7 de servicio al cliente.
* Los recursos deben ser escalables y elásticos al tiempo.
* Altos niveles de disponibilidad de infraestructura – 99.9%
* Mínimo tiempo de respuesta a las peticiones

# Quality

La arquitectura contribuye en la calidad de diversas características en la aplicación principalmente en la parte funcional. Definiendo unas tácticas para el aseguramiento de los atributos de calidad que se basaron en las encontradas en el libro Software Architecture in practice.

* Seguridad: La seguridad es un aspecto importante en el desarrollo de todos los proyectos de software, ya que la respuesta a los ataques y la recuperación de estos es un punto vital para el correcto funcionamiento de la aplicación. Para el aseguramiento de este atributo de calidad se utilizaran algunas tácticas arquitectónicas como lo son la autenticación, la autorización de usuarios y mantener la confidencialidad de los datos, estas tácticas se aplicaran al proyecto a través de la utilización de funcionalidades proporcionadas por el framework utilizado (Yii).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo de Calidad** | Seguridad | | | |
| **Interés del atributo** | Protección de datos y manejo de usuarios | | | |
| **Descripción** | El sistema debe tener alta seguridad para proteger tanto los datos de los usuarios como de proyectos, investigaciones, grupos y afines que requieren cierto grado de confidencialidad, además del manejo de usuarios para establecer un dominio de la información con la que va a interactuar. | | | |
| **ID Escenario** | 001 | | | |
| **Escenario** | Al ingresar al sistema el usuario hace el log in y el sistema le muestra sus opciones disponibles dependiendo de su rol, y no tendrá opción de ingresar de ningún modo a datos de otros dominios. | | | |
|  |  |  |  |  |
|  |  | **Fuente del Estimulo** | El usuario |  |
|  |  | **Estímulo** | El usuario pueda usar el sistema para informarse y llevar a cabo sus tareas |  |
|  |  | **Artefacto** | Página web de log in y home |  |
|  |  | **Ambiente** | Sistema comunidad Colciencias |  |
|  |  | **Respuesta** | El usuario queda satisfecho al encontrar todas las funcionalidades que necesita después de ingresar al sistema. |  |
|  |  | **Medida de respuesta** | Ingresa los datos correctamente y el sistema habilita el home en menos de 3 segundos |  |
|  |  |  |  |  |
| **Importancia** | Alta | | | |

* Usabilidad: La usabilidad es uno de los aspectos más trascendentales en el desarrollo de un proyecto que va dirigido a una comunidad o un grupo de usuarios debido a que este es el factor determinante de la aceptación por parte de estos a utilizar la aplicación. Para el aseguramiento de este atributo de calidad se aplicaran dos técnicas arquitectónicas las cuales son mantener el modelo de tareas, ya que este está plenamente definido en el módulo a desarrollarse y la separación de la interfaz de usuario del resto de la aplicación utilizando el modelo vista controlador. El desarrollo de la interfaz y la implementación del modelo vista controlador se realizaran utilizando las distintas herramientas que el framework Yii facilita para la elaboración eficaz de estas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo de Calidad** | Usabilidad | | | |
| **Interés del atributo** | El sistema maneja alta cantidad de información de interés para cada usuario, por ende una buena usabilidad hará que la experiencia de usuario sea la mejor y que el usuario acepte el sistema fácilmente. | | | |
| **Descripción** | El sistema en su interacción con el usuario debe de brindar una interfaz intuitiva para optimizar al máximo la disposición de La información. | | | |
| **ID Escenario** | 002 | | | |
| **Escenario** | Al ingresar al sistema el usuario encuentra una interfaz amigable e intuitiva con información de interés general, al hacer el log in el sistema le habilita el home en el cual encontrara una interfaz amigable y con una buena disposición de la información de interés específico del usuario. | | | |
|  |  |  |  |  |
|  |  | **Fuente del Estimulo** | El usuario |  |
|  |  | **Estímulo** | El usuario encuentra la información de manera adecuada e intuitiva |  |
|  |  | **Artefacto** | Todas las páginas de la comunidad |  |
|  |  | **Ambiente** | Sistema comunidad Colciencias |  |
|  |  | **Respuesta** | El usuario queda satisfecho al encontrar toda la información de manera adecuada y amigable. |  |
|  |  | **Medida de respuesta** | En todo momento el sistema brinda interfaz amigable e intuitiva |  |
|  |  |  |  |  |
| **Importancia** | Alta | | | |

* Testiabilidad: Es la parte del proceso de software que más recursos puede tomar, ya que la aplicación de un buen modelo de pruebas permite mostrar la calidad que posee el software antes de que este se utilice en un ambiente real disminuyendo la probabilidad de aparición de fallos. Para el aseguramiento de este atributo de calidad se aplicaran la táctica de la administración de las entradas y salidas separando la interfaz de la implementación, esto permitirá una fácil inclusión de nuevas funcionalidades para probar y la sustitución de los diferentes módulos, también la utilizará la táctica de Built-in monitors que permite el monitoreo del funcionamiento de la aplicación mostrando de una manera más completa el comportamiento que está teniendo el software con las diferentes entradas proporcionadas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo de Calidad** | Testiabilidad | | | |
| **Interés del atributo** | Garantizar el correcto funcionamiento de los requerimientos funcionales | | | |
| **Descripción** | El sistema debe garantizar el correcto funcionamiento de los requerimientos funcionales, para ello es primordial que el sistema este diseñado de manera que se le puedan hacer pruebas y mitigar los errores. | | | |
| **ID Escenario** | 003 | | | |
| **Escenario** | En las diferentes etapas del desarrollo del aplicativo se debe probar la correcta funcionalidad de los requerimientos funcionales. | | | |
|  |  |  |  |  |
|  |  | **Fuente del Estimulo** | tester |  |
|  |  | **Estímulo** | Fácil diseño de pruebas |  |
|  |  | **Artefacto** | Los archivos php |  |
|  |  | **Ambiente** | Sistema comunidad Colciencias |  |
|  |  | **Respuesta** | El tester realiza las pruebas fácilmente debido a la organización del código |  |
|  |  | **Medida de respuesta** | En todo momento el sistema brinda interfaz amigable e intuitiva |  |
|  |  |  |  |  |
| **Importancia** | Alta | | | |

* Modificabilidad: Es la posibilidad de aplicar cambios a la aplicación en cualquiera de los aspectos de esta que puede ir desde el cambio de color de la interfaz hasta incluso la modificación de un módulo funcional completo, estos cambios toman tiempo, esfuerzo y genera un costo para los encargados del desarrollo del proyecto por lo cual se debe tener especial cuidado. Para el aseguramiento de este atributo de calidad se utilizaran tácticas que previenen el efecto dominó manteniendo interfaces existentes y agregando intermediarios que permitirán la utilización de la información proporcionada por los diferentes módulos y la pasara a los que necesiten esta para poder realizar sus funcionalidades aun teniendo sintaxis de información diferentes, utilizando el soporte brindado por el lenguaje de programación PHP y el framework Yii.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo de Calidad** | Modificabilidad | | | |
| **Interés del atributo** | Modificar o adicionar funcionalidades fácilmente. | | | |
| **Descripción** | El sistema debe seguir una arquitectura que permita un relativamente fácil acoplamiento de nuevas funcionalidades o la modificación de una de ellas sin afectar severamente todo el sistema. | | | |
| **ID Escenario** | 004 | | | |
| **Escenario** | Cualquier cambio en las reglas de negocio solo afectaría el controlador y posiblemente un pequeño cambio en la vista, un cambio en la vista no afectaría ningún otro componente, siendo asi el MVC el patrón arquitectónico más ideal para conllevar posibles cambios. | | | |
|  |  |  |  |  |
|  |  | **Fuente del Estimulo** | Grupo de trabajo |  |
|  |  | **Estímulo** | Fácil acoplamiento o cambio |  |
|  |  | **Artefacto** | Módulos del sistema |  |
|  |  | **Ambiente** | Sistema comunidad Colciencias |  |
|  |  | **Respuesta** | El equipo de trabajo no debe desechar muchos recursos hechos |  |
|  |  | **Medida de respuesta** | En cualquier etapa del proyecto se pueden acoplar o cambiar funcionalidades sin mayor pérdida de recursos. |  |
|  |  |  |  |  |
| **Importancia** | Alta | | | |