\*

***Documento arquitectura de software***

**1.** **Historia de revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Autor |
| 18/11/2019 | 1.0 | Creación | Sebastián Jiménez |
|  |  |  |  |

**2. Arquitectura de software**

**2.1. Introducción**

Este documento es actualizable de acuerdo con los cambios técnicos de arquitectura que vayan apareciendo. Se detalla el equipo que mantiene la infraestructura técnica y los requerimientos de cambios que el equipo pudiera tener. El objetivo del documento es mantener organizada la Arquitectura Técnica Organizacional y proveer una fuente de referencia para los analistas y diseñadores de la aplicación.

El objetivo general de la creación de los requisitos de gestión de personas es para mejorar la visualización de cada uno de los datos de personas registradas, ya sea para elegir la mejor opción a la hora de hacer un crédito o para llevar un control de cada una de las personas.

**2.2. Descripción general del sistema**

**2.2.1. Proceso**

Programa

Esquemas de diseño

Registro

Aprobar

Verificar personal

Inicio de sesión

El sistema estará constituido por los siguientes componentes:

* Inicio de sesión
* Registro
* Verificar personal
* Aprobar

**2.3. Definición de la arquitectura**

La arquitectura seleccionada para el desarrollo del sistema es una arquitectura n-capa con cliente WEB.

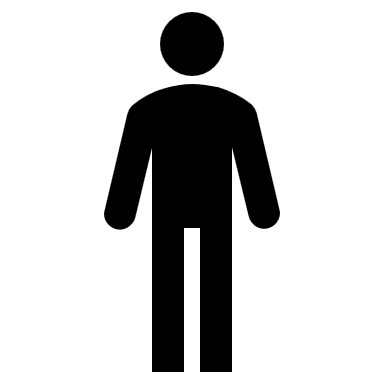
Identificamos como Usuarios comunes aquellos que acceden al sistema para hacer uso de la funcionalidad de la aplicación, e identificamos como usuarios administradores del sistema a aquellos que acceden para realizar mantenimiento de la aplicación, administrar seguridades, catálogos, entre otros.

Ambos tipos de usuarios dispondrán de una aplicación cliente web, que se comunicará con una aplicación de servidor la cual implementará las reglas del negocio y a su vez será la de intermediaria entre la aplicación cliente web y la base de datos.

**2.4. Vista de casos de uso**

La vista de casos de uso se encuentra dentro de la carpeta de este proyecto

**2.4.1. Seguridades**

****

* Registro de Usuarios: Permite a un usuario no registrado, registrarse para hacer uso del sistema
* Autenticación: Permite a un usuario autenticarse para usar el sistema
* Regresar: Permite a un usuario registrado, regresar de la página en la que se encuentre.
* Inicio: Permite a un usuario autenticado ingresar a la página principal.

**2.5. Arquitectura de infraestructura**

**2.5.1. Plataforma de software de base**

Cliente:

|  |  |
| --- | --- |
| Sistema Operativo | Multiplataforma |
| Browser | Internet Explorer, Chrome, Firefox |
| Tecnologías  utilizadas | Angular  HTML5  CSS3 |
| Lenguajes utilizados | TypeScript  Java |

Servidor de aplicación:

|  |  |
| --- | --- |
| Sistema Operativo | Multiplataforma |
| Application Server | Tomcat |
| Lenguajes utilizados | Java J2EE |
| Frameworks utilizados | Springboot  Hibernate |

Servidor de bases de datos:

|  |  |
| --- | --- |
| Sistema Operativo | Multiplataforma |
| Base de datos | MySQL |

**2.6. Diagrama de despliegue**

Pretendemos a través de este diagrama mostrar la arquitectura de ejecución del programa.

El diagrama representa cómo y dónde se desplegará el sistema.

A continuación, se muestra el diagrama de Despliegue de la aplicación:

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Las máquinas físicas se representan como nodos.

El nodo es un elemento donde se ejecutan los componentes.

Los componentes embebidos son aplicaciones, librerías y motor de base de datos.

Entre los nodos y componentes se comunican a través de relaciones lo que indica el tipo de conexión entre ellos.

Toda la aplicación estará montada en un único servidor en el cual se encontrará procesando el servidor de aplicaciones (Tomcat) y la base de datos (MySQL 5.5)

**2.7. Diagrama de componentes**

El diagrama de Componentes muestra los módulos en los que estará dividida la funcionalidad del sistema.

Los conectores identifican la relación entre los mismos. La aplicación cuenta con módulos claramente definidos los cuales acceden al modelo de datos único de la solución.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Log4J: Framework para facilitar las tareas de escritura de logs.

Módulo Administrador: componente donde se concentra la lógica de negocio referente a las operaciones realizadas por usuarios Administrador.

Hibernate: es una herramienta que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación.

MySQL: Motor de Base de Datos seleccionado para la aplicación.

**2.8. Diagrama de clases**

**2.8.1. Seguridades**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**2.9. Modelo de datos**

**2.9.1. Seguridades**

**Imagen que contiene texto, mapa

Descripción generada automáticamente**

**2.10. Herramientas de desarrollo**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**2.10.1. Estándares de desarrollo**

**Nomenclatura**:

En general para los nombres se utilizará CamelCase, donde cada palabra comienza con una letra mayúscula, por ejemplo: ListaUsuarios.

**Paquetes**:

Se utilizará el siguiente estándar: com.<Empresa>.<Aplicacion>.<Modulo>.<Paquete>.<Clase> Por ejemplo: com.utp.edu.co.grupo

**Clases**:

Los nombres deben estar en CamelCase y en singular. Por ejemplo: class Usuario. **Métodos**:

Los nombres deben estar en mayúsculas y minúsculas. Usar verbos para describir lo que hace el método. Por ejemplo: int CalcularDimension ()

**Variables**:

Los nombres deben iniciar la primera letra en minúsculas; continuando deben en mayúsculas y minúsculas. Los nombres deben representar lo que el valor de la variable representa. Por ejemplo: string nombreUsuario