UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA: INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de: INGENIEROS DE SISTEMAS

TEMA:
ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DEL MÓDULO DE ADMINISTRACIÓN DEL GEOPORTAL DE LA COMUNIDAD SALESIANA VERSIÓN 2.0

AUTORES:
BYRON PATRICIO SANDOVAL ESPÍN
WASHINGTON OSWALDO TUTILLO TUTILLO

DIRECTOR:
GUSTAVO ERNESTO NAVAS RUILOVA

Quito, abril de 2015

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DEL TRABAJO DE GRADO

Nosotros, autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de titulación y su reproducción sin fines de lucro.

Además, declaramos que los conceptos, análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad de los autores.

Quito, Abril 2015.

---	---
Byron Patricio Sandoval Espín	Washington Oswaldo TutilloTutillo
CC: 171778074-4	CC: 172159996-5

AGRADECIMIENTO

A la Universidad Politécnica Salesiana por habernos brindado un espacio de estudio, en el cual logramos adquirir todos los conocimientos necesarios para lograr culminar con éxito la educación superior y con ello ser profesionales para el servicio de la sociedad.

A nuestro tutor de trabajo de grado Ing. Gustavo Navas, quien con su ayuda hemos logrado terminar este trabajo de grado y con ello culminar con éxito nuestra carrera universitaria.

A nuestros familiares quienes con su ayuda y aliento, nos han apoyado para lograr la culminación de nuestra carrera.

Byron Sandoval, Oswaldo Tutillo

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN

El presente producto se basa en la nueva versión del módulo de Administración del GEOPORTAL DE LA COMUNIDAD SALESIANA y en los nuevos requerimientos que el usuario ha realizado.

El presente documento contara de 5 capítulos que se detallan a continuación:

El capítulo uno “MARCO TEÓRICO” contiene información de la metodología XP y herramientas de desarrollo que se utilizaron para el presente trabajo.

El capítulo dos “REQUERIMIENTOS DEL SISTEMA” contiene los requerimientos del usuario de forma detallada para el módulo a desarrollar.

El capítulo tres “DISEÑO” contiene el análisis de los requerimientos para crear los diagramas de clases, base de datos, secuencia, entre otros que servirán como base para la creación del módulo.

El capítulo cuatro “DESARROLLO” contiene las funcionalidades desarrolladas para realizar la versión 2.0 del módulo de Administración del Geoportal Salesiano.

El capítulo cinco “IMPLEMENTACIÓN” contiene los pasos necesarios para implementar la nueva versión del módulo.

ABSTRACT

This product is based on the new version of Release Management GEOPORTAL of the Salesian community and the new requirements that the user has made.

This document count of five chapters below:

Chapter one "THEORETICAL" contains the XP methodology and development tools that were used for this study.

Chapter two "SYSTEM REQUIREMENTS" contains user requirements in detail for the module to be developed.

Chapter three "DESIGN" contains the analysis of the requirements for creating class diagrams, database, string, etc. that will serve as the basis for creating the module.

Chapter four "DEVELOPMENT" contains the features developed for version 2.0 of the Administration module Salesian Geoportal.

Chapter five "IMPLEMENTATION" contains the necessary steps to implement the new version of the module.

INTRODUCCIÓN

La Inspectoría Salesiana “Sagrado Corazón de Jesús”, nace el 31 de julio de 1973 gracias al Decreto del Rector Mayor. Es producto de la unificación de la Inspectoría de Quito con la de Cuenca, es así que se crea la Inspectoría del Ecuador con sede en Quito y además se establece una Delegación para el Vicariato de Méndez.

Y a mediados del año 2002, en la calle Madrid E12-39 y Andalucía, se puso la primera piedra de la nueva sede Inspectorial, bendecida por el P. Esteban Ortiz y en presencia de numerosos miembros de la Familia Salesiana. La nueva casa está ubicada en la Calle Madrid E12-68 y Andalucía.

El 26 de febrero de 2004 se inauguró la Casa Inspectorial, con la presencia del Rector Mayor de los Salesianos; Pascual Chávez. Actualmente existen más de 26 casas salesianas y 173 salesianos: 129 sacerdotes; 16 coadjutores; 20 salesianos clérigos y 8 novicios.(Salesianos, 2014)

La expansión cada vez mayor de las casas y obras salesianas en el Ecuador han evidenciado que el actual Geoportal con el que cuenta la Comunidad Salesiana necesita de mejoras en cada uno de los módulos que lo conforma. Sobre todo en el módulo de administración, ya que por la gran cantidad de información necesita de adecuaciones para el ingreso de información y sobre todo en el aspecto de seguridad de la información.

Es por ello que se ha analizó este problema y se construyó la versión 2.0 del módulo de administración, con el objetivo de mitigar estas vulnerabilidades y sobre apoyar en la administración de tan grande y diverso sistema como es el Geoportal de la Comunidad Salesiana.

CAPÍTULO 1
MARCO TEÓRICO

1.1 El Objeto de la investigación.

El objeto de la investigación del trabajo de tesis, busca construir la versión 2.0 del módulo de Administración del Geoportal de la Comunidad Salesiana; así como mejorar y/o añadir funcionalidades que permitan optimizar el módulo y la interacción con el usuario.

El desarrollo del trabajo de grado, ofrece a los usuarios del Geoportal de la Comunidad, un módulo de administración funcional y de fácil uso; brindando una herramienta que permita administrar de manera eficiente los diferentes recursos de los que disponga el sistema geográfico. Una parte importante del trabajo de grado es optimizar el manejo de los datos geográficos por medio del formato ligero de intercambio de datos geográficos GeoJSON.

1.2 Planteamiento del problema.

El proyecto, busca dar continuidad a la tesis que desarrollo el Módulo de administración con los submódulos: Gestión de la información de la organización y Gestión de datos geográficos en el Geoportal de la Comunidad Salesiana; así como mejorar y/o añadir funcionalidades que permitan optimizar el módulo y la interacción con el usuario.

Por otro lado, la funcionalidad de los subcomponentes del módulo de administración versión 1.0; limita las tareas a los usuarios a las que se detallan a continuación.

- **Gestión de información de la organización.**
  Permite gestionar (ingresar, actualizar, eliminar) la información: casas salesianas, obras, lugares, beneficiarios, tipos de beneficiarios, colaboradores, área pastoral; en la base de datos del Geoportal de la Comunidad Salesiana.

- **Gestión de información geográfica.**

  Ingresa la ubicación de las obras salesianas por medio de la longitud y latitud, actualmente estos datos no pueden ser comprobados dando lugar a inconsistencia de información que posteriormente se reflejarán en el visualizador de datos espaciales.

Además, la carga de datos de la ubicación de los beneficiarios, se realiza por medio de archivos shape. Los archivos shape, son procesados a través del software Quantum GIS tomando como base los datos recopilados con dispositivos GPS; antes de ser almacenado en la base de datos principal.

1.3 Objetivos

1.3.1 Objetivo general.

Analizar, diseñar y construir el módulo de Administración del Geoportal de la Comunidad Salesiana versión 2.0, integrando nuevas características y funcionalidades al submódulo de gestión de la información y al submódulo de gestión de datos geográficos.

1.3.2 Objetivos específicos.

1. Realizar el levantamiento de la información, de la casa Salesiana ubicada en la ciudad de Manta, provincia de Manabí.
2. Utilizar dispositivos GPS, para la captura de trazas y datos específicos del área designada.
3. Analizar la estructura de la base de datos actualmente usada por el módulo de administración y realizar la optimización de la misma.
4. Definir las capas, clases e interfaces necesarias para el desarrollo del módulo de administración.
5. Diseñar la nueva versión del módulo de administración, utilizando metodología XP y las herramientas UML (Unified Modeling Language).
6. Implementar las pruebas necesarias, contempladas en la metodología XP.

7. Realizar la implementación de módulo de administración del Geoportal en los servidores de Centro de Investigación y Modelamiento Ambiental – CIMA.
8. Crear las facilidades para el proceso de integración del Geoportal de la Comunidad Salesiana.

1.4 Alcance.

El resultado final de este proyecto será la obtención de la versión 2.0 del módulo de Administración del Geoportal Salesiano con las siguientes características:

- Implementación de seguridad para el módulo de administración.
- Implementación de un menú dinámico para el ingreso al módulo.
- Creación de perfiles de usuarios y permisos de acceso.
- Creación de la opción para generar y posterior descarga de archivos en formato GeoJSON.
- Creación de una nueva interfaz para el ingreso de polígonos como área de influencia para las obras salesianas.

1.5. Justificación.

El presente proyecto, busca dar continuidad al proyecto de tesis que implementa: el Módulo de gestión de la información de la organización y el Módulo de gestión de datos geográficos de la Universidad Politécnica Salesiana; así como mejorar y/o añadir las funcionalidades de seguridad, ingreso de información y visualización que actualmente se encuentran implementadas en la versión anterior.

Actualmente en este módulo no se encuentra implementada la funcionalidad de seguridad, peor aún la funcionalidad de perfiles; por esta razón los usuarios pueden ver información que no es concerniente a ellos. Por este inconveniente se hace necesario que se implemente la funcionalidad de seguridad integrando perfiles a este módulo, permitiendo así discernir la información y que esta sea entregada de acuerdo a la necesidad de cada uno de los usuarios.

Con la implementación del formato GeoJSON, se prevé reducir el tamaño de archivos y así disminuir el peso del sistema, optimizando los recursos y mejorando el rendimiento del mismo.

Además de esta funcionalidad, se busca mejorar el módulo de Administración del Geoportal de la Inspectoría de la Comunidad Salesiana implementando las principales modificaciones que se citan a continuación:

- Manejo de tiempos de sesión por usuarios.
- Cambios e implementación de métodos de carga de datos espaciales en la base de datos.
- Administración de parámetros de diseño general y presentación de datos espaciales.
- Edición de la información general de las obras salesianas.

La Versión 2.0, del módulo de Administración, será desarrollada para uso de la Comunidad Salesiana; permitiendo administrar el Geoportal y las características de presentación de los datos en el visualizador. El módulo de administración, posee características que serán administradas de manera eficiente; para que de esta manera se pueda obtener acceso a la información de las casas y de obras Salesianas.

Además, el desarrollo total del sistema en código abierto, permite un proceso de mejora continua y la posibilidad de seguir actualizando el sistema según sea el requerimiento de la Comunidad Salesiana en versiones posteriores.

1.6. Justificación teórica.

Debido al crecimiento de las comunidades salesianas, el Geoportal Salesiano necesita mejorar algunos de sus módulos para solventar las necesidades que se están presentando, es así que uno de los módulos a mejorar es el de Administración, el cual necesita incorporar nuevas funcionalidades.

Las necesidades que necesita son especialmente en la seguridad, administración de usuarios y sobre todo en la generación de archivos en formatos que son necesarios para el Visualizador del Geoportal.

Con estas funcionalidades se esperar mejorar el rendimiento del Geoportal y sobre todo solventar las necesidades que presenta actualmente el Geoportal Salesiano.

1.7. Justificación práctica.

De acuerdo con los objetivos de estudio el cual indica entre otros, diseñar la nueva versión del módulo de Administración, utilizando metodología XP y las herramientas UML (Unified Modeling Language); se busca desarrollar nuevas funcionalidades en el módulo de Administración del Geoportal de la Inspectoría de la Comunidad Salesiana a la versión 2.0, que actualmente presenta falencias principalmente en el aspecto de seguridad y velocidad de despliegue de este módulo.

Con las implementaciones previstas para esta nueva versión se puede contar con un módulo de administración más robusto, que integre seguridad y con una mayor velocidad para la que se tiene actualmente.

1.8 Marco teórico.

Geoportal

> El Geoportal es un sitio web cuya finalidad es ofrecer a los usuarios el acceso a una serie de servicios y recursos como imágenes, fotografías y mapas basados en la información geográfica. Permite el descubrimiento, el acceso y la visualización de los datos geoespaciales, utilizando un navegador estándar, y posibilita la iteración y el intercambio de información entre los usuarios mediante los datos georeferenciados (Barcelona, Geoportal bcn).

Georeferenciación: “Básicamente es una técnica geográfica, que consiste en asignar mediante cualquier medio técnico apropiado, una serie de coordenadas geográficas

procedentes de una imagen de referencia conocida, a una imagen digital de destino" (Fernández, 2013).

GeoJSON: "Es un formato de intercambio geoespacial basado en JSON (Java Script Object Notation) y permite codificar varias estructuras geográficas" (Gracia, 2013).

JSF: "La tecnología Java Server Faces es un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java" (Castellano, 2013).

GPS: "En síntesis podemos definir el GPS como un Sistema Global de Navegación por Satélite (GNSS) que nos permite fijar a escala mundial la posición de un objeto, una persona, un vehículo o una nave" (EURORESIDENTES, 2013).

Base de datos: "Es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. Una base de datos es un sistema de archivos electrónico" (MASADELANTE, 2013).

Programación por capas: "Es un estilo de programación, su objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos" (Calle, 2013).

Capa de presentación: "Esta capa es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso" (Calle, 2013).

Capa de negocio: "Aquí es donde, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse" (Calle, 2013)**.**

Capa de datos:

> Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio (Calle, 2013).

Base de datos espaciales:

> Las bases de datos espaciales son bases de datos que almacenan datos espaciales, o en otras palabras, los datos relacionados a los espacios en el mundo físico, las partes de los organismos vivientes, el diseño en ingeniería y muchos otros espacios de interés. Los datos en una base de datos espacial a menudo son capturados inicialmente en forma de imágenes digitales, por lo que las bases de datos espaciales algunas veces son llamadas pictóricas o de imágenes (ehowenespano, 2013).

Seguridad informática: "La seguridad informática es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático. De todas formas, no existe ninguna técnica que permita asegurar la inviolabilidad de un sistema"(DEFINICIONDE, 2013).

Metodología XP: "La programación extrema es una metodología de desarrollo ligero (o ágil) basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas"(procesos de software, 2013).

1.9. Metodología Extreme Programming.

Como se especificó en el anterior apartado, la metodología XP es una metodología ágil basada en una serie de valores y prácticas de las cuales especificamos las más relevantes para nuestro proyecto:

- Planificación incremental

  El objetivo de la XP es generar versiones de la aplicación tan pequeñas como sea posible, pero que proporcionen un valor adicional claro, desde el punto de vista del negocio. A estas versiones se las denomina **releases**.

  Una release cuenta con un cierto número de historias. La historia es la unidad de funcionalidad en un proyecto XP, y corresponde a la mínima funcionalidad posible que tiene valor desde el punto de vista del negocio(OFBiz, 2014).

- Testing

  La ejecución automatizada de tests es un elemento clave de la XP. Existen tests internos, que garantizan un correcto funcionamiento.

  El cliente es el responsable de definir los tests de aceptación, no necesariamente de implementarlos. Él es la persona mejor cualificada para decidir cuál es la funcionalidad más valiosa(OFBiz, 2014).

- Diseño simple

  Otra práctica fundamental de la Programación Extrema es utilizar diseños tan simples como sea posible. El principio es "utilizar el diseño más sencillo que consiga que todo funcione"(OFBiz, 2014).

- Propiedad colectiva del código

  La XP aboga por la propiedad colectiva del código. En otras palabras, todo desarrollador tiene autoridad para hacer cambios a cualquier código, y es responsable de ellos. Esto permite no tener que esperar de otros cuando todo lo que hace falta es algún pequeño cambio(OFBiz, 2014).

- Integración continúa

  En muchos casos la integración de código produce efectos laterales imprevistos, y en ocasiones la integración puede llegar a ser realmente traumática, cuando dejan de funcionar cosas por motivos desconocidos. La Programación Extrema hace que la integración sea permanente, con lo

que todos los problemas se manifiestan de forma inmediata, en lugar de durante una fase de integración más o menos remota(OFBiz, 2014).

- Cliente en el equipo

  Algunos de los problemas más graves en el desarrollo son los que se originan cuando el equipo de desarrollo toma decisiones de negocio críticas. Esto no debería ocurrir, pero en la práctica, esto ocurre con frecuencia no se obtiene feedback del cliente con la fluidez necesaria. La XP intenta resolver este tipo de problemas integrando un representante del negocio dentro del equipo de desarrollo. Ésta persona siempre está disponible para resolver dudas y para decidir qué se hace y qué no se hace en cada momento, en función de los intereses del negocio. (OFBiz, 2014).

- Releases pequeñas

  Siguiendo la política de la XP de dar el máximo valor posible en cada momento, se intenta liberar nuevas versiones de las aplicaciones con frecuencia. Éstas deben ser tan pequeñas como sea posible, aunque deben añadir suficiente valor como para que resulten valiosas para el cliente(OFBiz, 2014).

- Estándares de codificación

  Para conseguir que el código se encuentre en buen estado y que cualquier persona del equipo pueda modificar cualquier parte del código es imprescindible que el estilo de codificación sea consistente. Un estándar de codificación es necesario para soportar otras prácticas de la XP(OFBiz, 2014).

CAPÍTULO 2
REQUERIMIENTOS DEL SISTEMA

En el presente capítulo, se describen los requerimientos a incorporarse en la versión 2.0 del módulo de Administración del Geoportal Salesiano. La implementación de estos requerimientos, tienen como finalidad mejorar las funcionalidades existentes en la versión 1.0 e incorporar nuevas funcionalidades; disponiendo de un módulo con funcionalidades útiles para los usuarios finales. Las funcionalidades contempladas en la nueva versión se detallan a continuación.

- Framework de seguridad para la aplicación.
- Creación de perfiles de usuario y permisos de acceso.
- Seguridad de usuarios.
- Acceso de información por perfiles.
- Implementación de un menú dinámico.
- Creación y descarga de archivos en formato GeoJSON (*.geojson).
- Creación de la funcionalidad para trazado de polígonos como área de influencia de una obra salesiana.
- Reportes en formato PDF (*.pdf).

2.1 Cuadro comparativo entre versiones.

Para mostrar más detalladamente cuales fueron los requerimientos de cada una de las versiones del módulo de Administración del Geoportal Salesiano, se hace necesario realizar un cuadro comparativo entre las versión 1 y la versión 2.

En la tabla 1, cuadro comparativo, se evidencia las diferencias entre versiones y la razón de ser del presente proyecto de tesis; además de indicar las nuevas funcionalidades que tendrá esta versión, que busca potenciar el módulo de Administración y por ende al Geoportal Salesiano.

Tabla 1. *Cuadro comparativo de los requerimientos entre las versiones 1 y 2 del módulo de Administración del Geoportal Salesiano.*

Requerimientos	Versión 1	Versión 2
Gestión de casas salesianas	Se creó la funcionalidad	Se utilizo de la versión anterior
Gestión de obras salesianas	Se creó la funcionalidad	Se utilizo de la versión anterior
Gestión de tipos de obras salesianas	Se creó la funcionalidad	Se utilizo de la versión anterior
Gestión de lugares	Se creó la funcionalidad	Se utilizo de la versión anterior
Gestión de colaboradores	Se creó la funcionalidad	Se utilizo de la versión anterior
Gestión de tipos de colaboradores	Se creó la funcionalidad	Se utilizo de la versión anterior
Gestión de beneficiarios	Se creó la funcionalidad	Se utilizo de la versión anterior
Gestión de ubicación de colaboradores	Se creó la funcionalidad	Se utilizo de la versión anterior
Framework de seguridad	No dispone	Se implemento la funcionalidad
Menú dinámico	Se dispone un menú estático básico	Se creó la funcionalidad para el menú
Gestión de usuarios	No dispone	Se creó la funcionalidad
Gestión de acceso a usuarios	Se dispone un acceso básico	Se creó la funcionalidad para el acceso
Gestión de perfiles de usuario	Se dispone una gestión básica no administrada por base de datos	Se potencio la funcionalidad de la versión anterior con uso de la base de datos.
Generación de archivos Geojson	No dispone	Se creó la funcionalidad
Creación de tabla en base de datos con el resultado de un query en el submódulo generador de archivos Geojson.	No dispone	Se creó la funcionalidad
Descarga de archivos Geojson	No dispone	Se creó la funcionalidad
Gestión del área de influencia de una obra salesiana.	No dispone	Se creó la funcionalidad

Reporte por obra salesiana	No dispone	Se creó la funcionalidad
Reporte por Casa Salesiana	No dispone	Se creó la funcionalidad

Nota. Cuadro comparativo entre la versión anterior y la nueva versión propuesta.
Elaborado por: Sandoval Byron &Tutillo Oswaldo

Como se puede evidenciar, la versión anterior de este módulo se encargo de la creación de las funcionalidades para la gestión de la información inicial del Geoportal Salesiano, mientras que la nueva versión se enfoca en la mejora e implementación de funcionalidades que se evidenciaron como necesarias a medida que se seguía utilizando este módulo en el Geoportal Salesiano. Es así que las funcionalidades de esta versión, estarán enfocadas hacia la seguridad y potenciar la administración de la información del Geoportal.

2.2 Requerimientos.

En los siguientes apartados, se describe cada uno de los requerimientos de esta nueva versión.

Para revisar las historias de usuario de cada uno de estos requerimientos, ver **Anexos 1, 2, 3 y 4**.

2.2.1 Framework de seguridad para la aplicación.

Para garantizar que los recursos de la aplicación estén seguros, es necesario que se implemente un mecanismo de seguridad, framework; que impida que usuarios no autorizados puedan acceder a funcionalidades e información del sistema que no le compete.

2.2.2 Creación de perfiles de usuarios y permisos de acceso.

Para gestionar la seguridad de los usuarios del sistema, es necesario que se pueda administrar la información y los permisos que tendrán cada uno de estos.

Siendo así, el módulo de Administración debe contar con la funcionalidad de creación, edición y eliminación de usuarios y además con la administración de permisos para cada uno de ellos.

2.2.3 Seguridad de usuarios.

Para esta parte es necesario mejorar y optimizar el acceso al sistema que mantiene actualmente el sistema, implementando el manejo de usuarios con sus respectivos perfiles y permisos; de acuerdo a la Casa Salesiana a la que pertenezca el usuario y tiempos de sesión activa en el sistema.

Para ello es necesario implementar las siguientes funcionalidades:

- Creación y edición de Usuarios.
- Creación y edición de Roles.
- Tiempo de sesión inactiva.
- Acceso a la información por perfiles.

En la tabla 2 se muestran los roles que dispone actualmente el módulo de Administración y los accesos que los mismos deben tener.

Tabla 2. *Roles y accesos al módulo de administración*

Roles	Acceso
Administrador	Administración de datos geográficos Administración de datos administrativos
Administrador GIS	Ingreso de datos geográficos. Ingreso de nuevas obras salesianas. Ingreso de áreas de influencia de las obras salesianas.
Administrador Datos	Ingreso y actualización de datos administrativos Ingreso de nuevos usuarios.

Nota. Descripción de los accesos que tiene cada uno de los Roles definidos para este módulo.

Elaborado por: Byron Sandoval & Oswaldo Tutillo

2.2.4 Implementación de un menú dinámico.

Para facilitar la administración del Geoportal Salesiano, es necesario la creación de un menú dinámico; permitiendo así agregar nuevos ítems al menú de una manera cómoda sin que sea necesario el desarrollo de otro menú, es por ello que esta funcionalidad debe ser configurable, administrable, escalable y que se adapte a los requerimientos de cada una de las obras salesianas.

2.2.5 Creación y descarga de archivos GeoJSON.

En vista que el Geoportal maneja una gran cantidad de datos y particularmente información geográfica, se necesita que esta información sea mostrada lo más rápido posible; para solventar las necesidades del usuario. Para mejorar el flujo de información, se ha pensado en utilizar el formato GeoJSON, para manejar la información de: casas y obras salesianas.

Por ello es necesario que el módulo de Administración cuente con una funcionalidad que permita generar archivos en formato .geojson; estos formatos son más livianos y permiten la visualización de los objetos georeferenciados más rápidamente.

2.2.6 Creación de una tabla en la base de datos, con el resultado de un query en el submódulo generador de archivos GeoJSON.

Como parte complementaria de la funcionalidad que permite generar archivos en formato geojson, se debe incorporar una utilidad para poder crear una tabla en la base de datos del Geoportal.

Esta utilidad debe de incluirse en la pantalla de la funcionalidad para generar archivos GeoJSON; para permitir crear la tabla con la estructura y datos seleccionados por el usuario. Además, se debe de personalizar el nombre con el cual se creara la nueva tabla; según sea definido por el usuario.

2.2.7 Creación de una nueva interfaz para el ingreso de polígonos como área de influencia de una obra salesiana.

Actualmente el módulo de administración cuenta con la funcionalidad para ingresar la ubicación de nuevas obras salesianas, mediante la georeferencia en un mapa interactivo; pero no se dispone de la funcionalidad para ingresar el área de influencia.

Es por eso que se ve necesario implementar una funcionalidad en el módulo de Administración, que permita georeferenciar el área de influencia de una obra salesiana; mediante una pantalla interactiva y de fácil uso. La funcionalidad debe permitir al usuario, trazar el área de influencia en un mapa y vincular el trazo a una obra salesiana seleccionada.

2.2.8 Reportes.

Como parte de la versión 2 del modulo de Administración, se debe incluir una funcionalidad que permita exportar archivos en formato PDF (*.pdf), según la selección del usuario. Esta funcionalidad debe permitir exportar los archivos si existe información en la base de datos.

La funcionalidad permite exportar archivos con información de:

- Casa Salesiana: exporta la información de todas las Obras Salesiana relacionas a una Casa Salesiana, seleccionada por el usuario.
- Obras salesiana: exporta la información de una Obra Salesiana, seleccionada por el usuario.

2.3 Especificación de casos de uso de los requerimientos.

Para una mejor comprensión de los requerimientos antes descritos se utilizan los diagramas de casos de uso que evidencia de mejor manera la iteración con los usuarios.

2.3.1 Definición de casos de uso, actores y relaciones:

- **Diagramas de caso de uso:**

  Los diagramas de caso de uso representa la forma en cómo un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso)(Salinas, 1996).

- **Actor:**

  Un actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema (Salinas, 1996).

En la tabla 3 se evidencia los actores y las tareas que desempeñan en el módulo de Administración.

Tabla 3. *Descripción de actores para el módulo de Administración*

Roles	**Tareas**
Administrador	Usuario que debe administrar tanto datos geográficos como administrativos.
Administrador GIS	Usuario que administra datos geográficos del geoportal
Administrador DATOS	Usuario que debe administrar datos administrativos del módulo.

Nota. Descripción de los roles que tiene el módulo y el acceso que tendrán en el sistema.

Elaborado por: Sandoval Byron & Tutillo Oswaldo

- **Caso de uso:**

  Es una operación o tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso(Salinas, 1996).

En la tabla 4 se especifica los casos de uso que serán implementados en esta versión.

Tabla 4. *Descripción de casos de uso del módulo de Administración versión 2.*

Caso de uso	**Tareas**
Menú dinámico	El usuario administrador puede crear o actualizar un nuevo menú
Creación y edición de usuarios y roles	El usuario administrador podrá crear, editar, eliminar un usuario y un rol o asignar un nuevo rol.
Creación y descarga de archivos GeoJSON	• El usuario administrador puede generar el query para consultar la información para el archivo geojson. • El usuario administrador puede generar el archivo en formato geojson. • El usuario Administrador puede descargar el archivo geojson. • El usuario Administrador puede generar una nueva tabla en la base de datos con la información filtrados en el proceso.

Nota. Descripción del caso de uso por cada nueva funcionalidad que se va a desarrollar.

Elaborado por: Sandoval Byron & Tutillo Oswaldo

2.4Escenarios.

"Es una interacción entre el sistema y los actores, que puede ser descrito mediante una secuencia de mensajes. Un caso de uso es una generalización de un escenario, para este caso se utilizara el rol de administrador".(Cáceres, 2011).

En la tabla 5 se evidencia los escenarios del módulo de administración versión 2, además de una breve descripción de cada uno de ellos.

Tabla 5. *Especificación de escenario del módulo de Administración geográfica*

Escenario	Descripción
Creación de un nuevo ítem para el menú dinámico	Permite la creación y edición de un nuevo ítem en el menú
Crear un nuevo usuario para el sistema	Permite la creación y edición de un nuevo usuario que tenga acceso al geoportal salesiano
Creación de un archivo GeoJSON	Permite la generación y descarga de un archivo GeoJSON con las coordenadas escogidas

Nota. Escenario del módulo de administración con las nuevas funcionalidades.

Elaborado por: Sandoval Byron & Tutillo Oswaldo

2.5 Especificación de casos de uso.

En el siguiente apartado se especificara cada uno de los casos de uso descritos anteriormente.

2.5.1 Especificación de casos de uso en la creación de un nuevo ítem para el menú dinámico.

Este caso de uso se encarga de la creación de un nuevo ítem para el menú, el cual ingresa en la base de datos a través del sistema.

En la tabla 6 se especifica el caso de uso para la creación un nuevo ítem para el menú dinámico.

Tabla 6. *Casos de uso del menú dinámico*

Caso de Uso # 1	Creación de un nuevo ítem para el menú dinámico
Actores	Usuario Administrador
Camino Principal	1. Crear el ítem del menú. 2. Escoger el perfil a cual será visible este submenú. 3. Brindar los permisos que tendrá el perfil escogido para ese submenú. 4. Guardar la información.
Precondiciones	Para crear el submenú tiene que tener perfil de administrador

Postcondiciones	El submenú se registra en la Bases de datos

Nota. Caso de uso para la funcionalidad creación de menú dinámico.

Elaborado por: Sandoval Byron &Tutillo Oswaldo

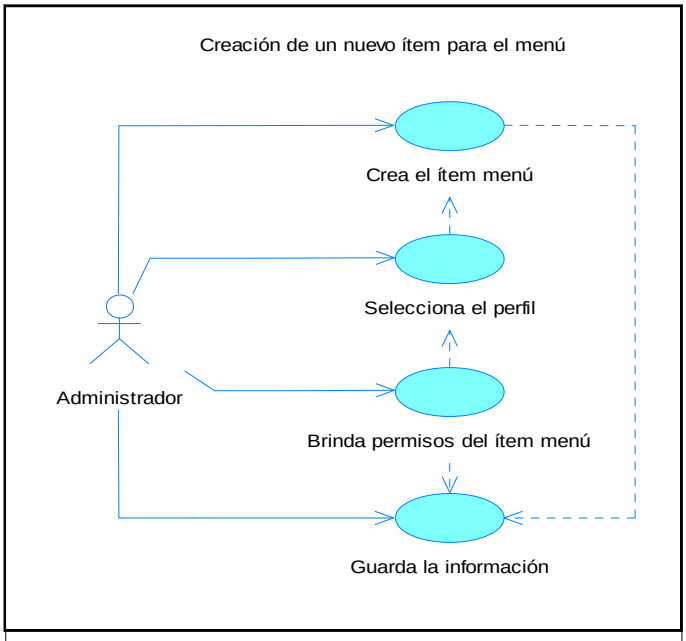


Figura 1. Caso de uso creación de un nuevo ítem menú

Elaborado por: Sandoval Byron &Tutillo Oswaldo

2.5.2 Especificación de casos de uso en la creación de un nuevo usuario para el sistema.

Este caso de uso se encargara de la creación de un nuevo usuario que pueda acceder al sistema, todos estos datos quedaran registrados en la base de datos.

En la tabla 7 se especifica el caso de uso para la creación de un nuevo usuario para el módulo de administración.

Tabla 7. *Casos de uso para la creación de un nuevo usuario para el módulo de Administración*.

Caso de Uso # 2	**Creación de un nuevo usuario para el sistema**
Actores	Usuario Administrador

Camino Principal	1. Crear el usuario y la contraseña para el usuario. 2. Asignar ese usuario a una persona que se encuentra registrada en el sistema. 3. Asignar el perfil que tendrá este usuario. 4. Guardar la información.
Precondiciones	Para crear el usuario el usuario en sesión debe tener perfil de administrador del sistema y el usuario no deberá estar ingresado anteriormente.
Postcondiciones	El usuario quedará registrado en la Bases de datos

Nota. Caso de uso para la creación de un nuevo usuario

Elaborado por: Sandoval Byron &Tutillo Oswaldo

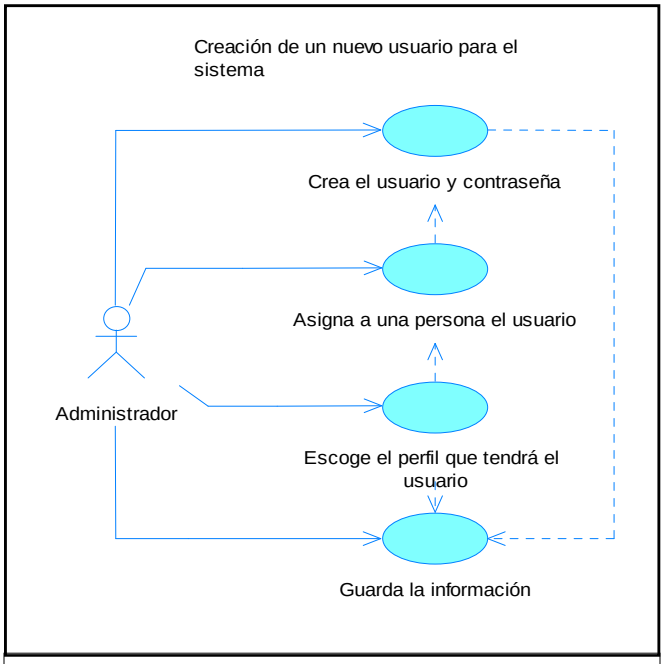


Figura 2. Caso de uso creación de un nuevo ítem menú

Elaborado por: Sandoval Byron &Tutillo Oswaldo

CAPÍTULO 3

DISEÑO

En el presente capítulo se da a conocer la estructura y el diseño de cada una de las pantallas, diagramas de clases, de base de datos y de secuencia que se incorporaron en esta versión.

Para realizar cada uno de los diseños se tomó en cuenta los requerimientos mencionados anteriormente, con la finalidad de indicar el resultado del módulo de administración con esta nueva versión, resaltando el manejo de datos, funcionalidad y comportamiento.

Para el diseño de la interfaz se inicio desde el esquema de la versión anterior, modificando algunos componentes que permiten, a la aplicación acoplarse con el usuario, de manera amigable y con un patrón minimalista es decir de menor carga visual para el usuario y que cumpla con los requisitos principales del usuario.

3.1. Modelo de la base de datos general.

El presente modelo de la base de datos permite visualizar la estructura inicial de la base de datos y la que se incremento para el desarrollo de esta nueva versión del módulo de administración del Geoportal Salesiano.

En la figura 3 y 4 se especifica el modelo conceptual y físico respectivamente, de la base de datos implementada en el trabajo de grado intitulado análisis, diseño e implementación del módulo de visualización y gestión de datos geográficos para el geoportal de la comunidad salesiana versión 2.0 desarrollado por los señores Cofre Víctor y Toledo Stalin, al cual se ha integrado las nuevas tablas creadas para el desarrollo de esta nueva versión, diferenciada en colores. La tabla tb_auditoria, que se encuentra sin relación, se utiliza para registrar las acciones del usuario en los diferentes submódulos.

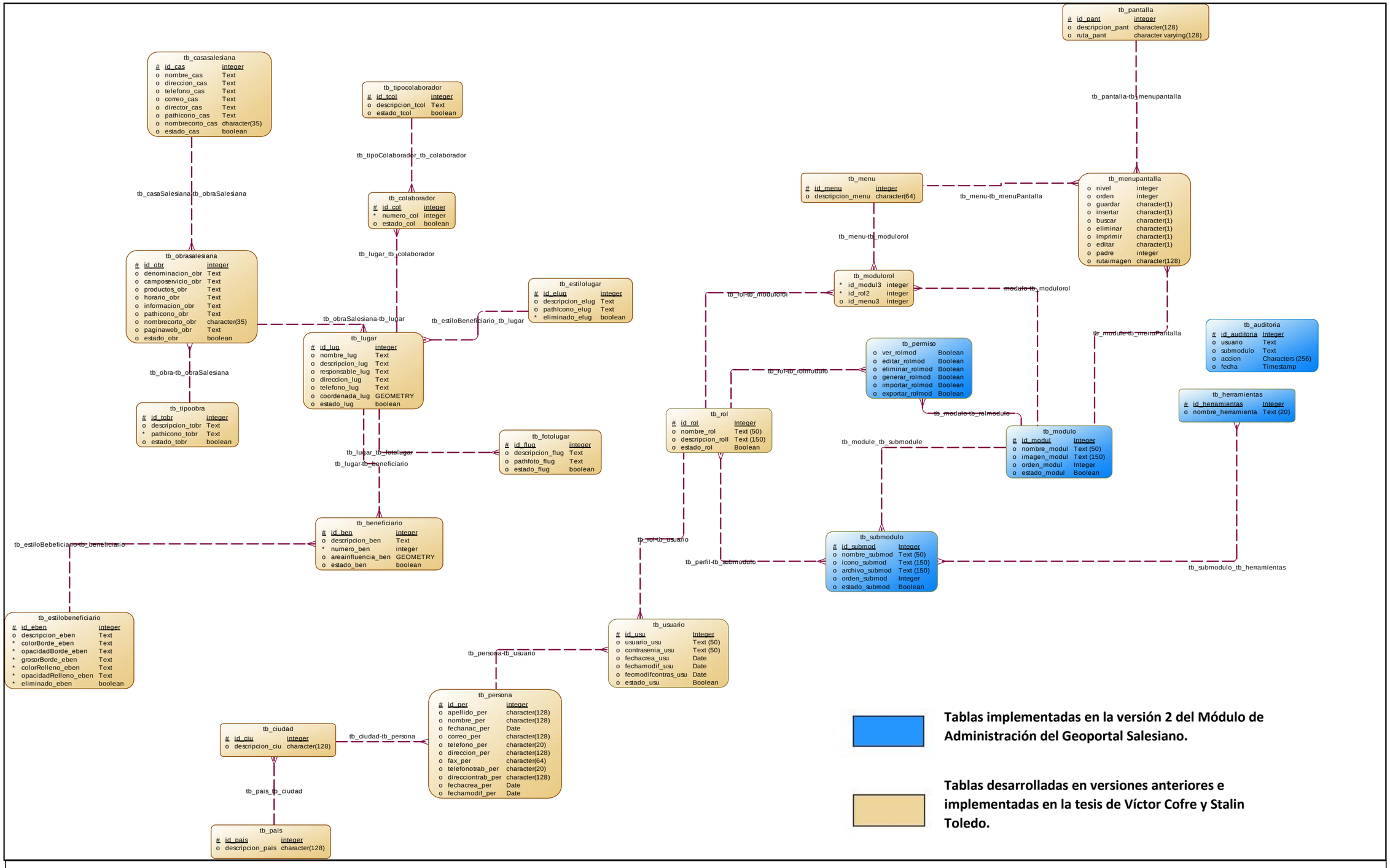


Figura 3. Modelo conceptual de base de datos

Elaborado por: Sandoval Byron & Tutillo Oswaldo

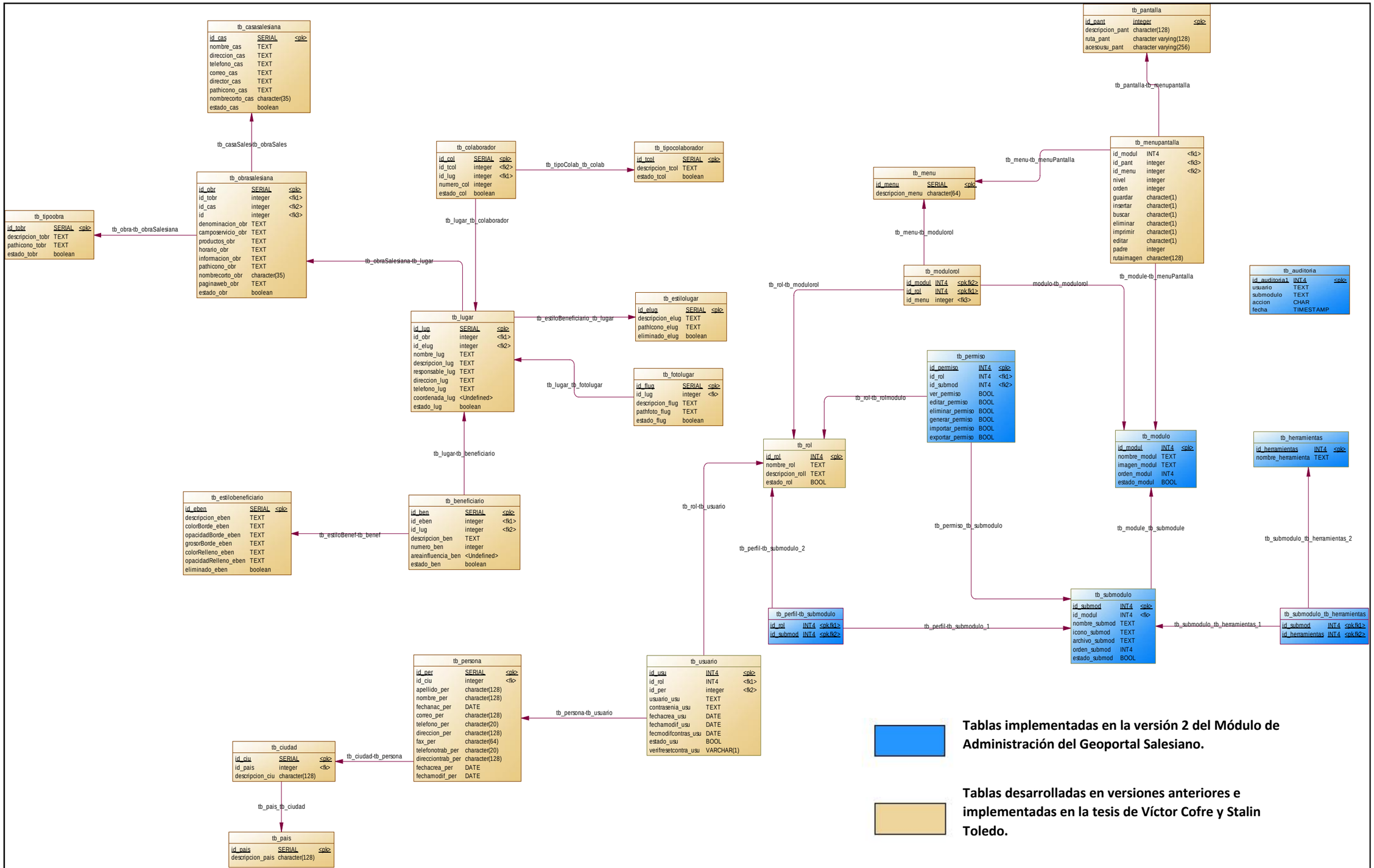


*Figura 4.*Modelo físico de base de datos diferenciado entre versiones

Elaborado por: Sandoval Byron & Tutillo Oswaldo

Como se pudo observar para realizar esta nueva versión, fue necesario implementar algunas tablas para la creación de las nuevas funcionalidades para esta versión.

En la figura 5 que se muestra a continuación se puede observar más detalladamente las nuevas tablas que fueron implementadas para realizar la versión 2 del módulo de Administración del Geoportal Salesiano.

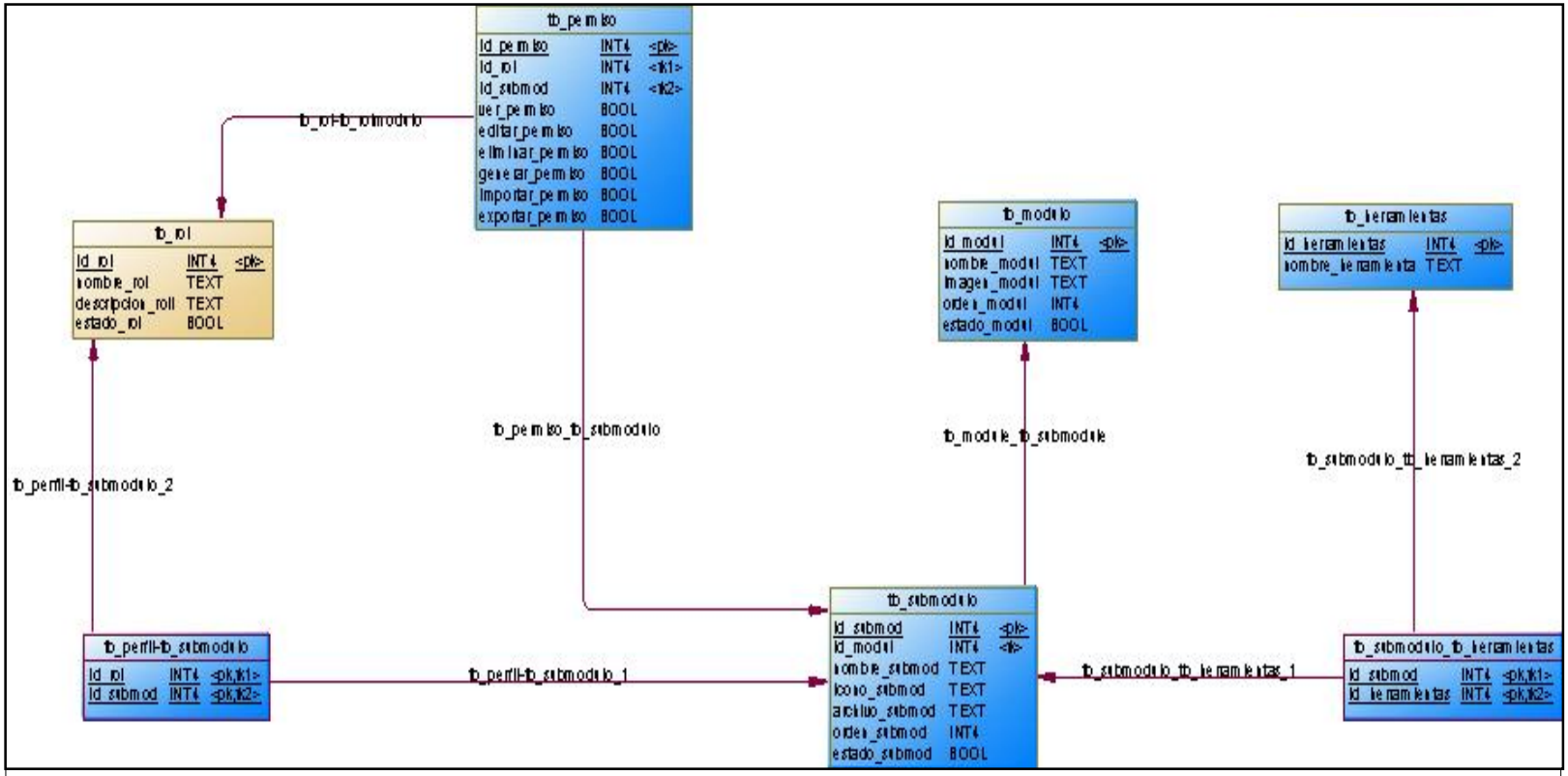


*Figura 5.*Modelo físico de la versión 2

Elaborado por: Sandoval Byron &Tutillo Oswaldo

3.2 Diccionario de base de datos.

En esta sección se muestra el diccionario de datos de las tablas creadas para el desarrollo de esta nueva versión; en ellas constan la información como el nombre de los campos, el tipo de dato, la descripción y el tipo de información que dicho campo almacenará.

Cabe indicar que para esta sección solo se tomó en cuenta las tablas que fueron incrementadas y modificadas para realizar la nueva versión del módulo de Administración del Geoportal Salesiano.

Cada uno de las tablas que se presenta a continuación cumple el objetivo de almacenar los datos necesarios para cada una de los formularios que se encuentran en esta nueva versión del módulo de Administración.

El diccionario de datos de las tablas implementadas en versiones anteriores; se las puede observar en el **Anexo 5, Diccionario de datos implementado en versiones anteriores.**

Tabla 8. *Diccionario de datos tabla pantalla*

Nombre	tb_pantalla			
Descripción	Almacena la información de las páginas del sistema, para su registro y posterior iteración con los datos necesarios por pantalla.			
Primary Key	Id_pant			
Key	**ColumnName**	**Data Type**	**NotNull**	**Description**
PK	id_pant	serial	Yes	Identifica la pantalla
	descripcion_pant	text	No	Descripción de la pantalla
	ruta_pant	text	No	Ruta de la pantalla
	acesousu_pant	text	No	Usuarios que tienen acceso a la pantalla.

Nota. Diccionario de datos tabla pantalla

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 9. *Diccionario de datos tabla país*

Nombre	tb_pais			
Descripción	Almacena los nombres de los países de origen de los usuarios registrados en el geoportal salesiano.			
Primary Key	id_pais			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_pais	serial	Yes	Código del país
	descripcion_pais	text	No	Nombre del país

Nota. Diccionario de datos tabla país.

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 10. *Diccionario de datos tabla ciudad*

Nombre	tb_ciudad			
Descripción	Almacena la información de las ciudades de origen los usuarios registrados en el geoportal salesiano.			
Primary Key	id_ciu			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_ciu	serial	Yes	Código ciudad
FK	id_pais	integer	Yes	Código del país
	descripcion_ciu	text	No	Nombre de la ciudad

Nota. Diccionario de datos tabla ciudad

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 11. *Diccionario de datos tabla persona*

Nombre	tb_persona			
Descripción	Almacena la información completa de las personas que interactuaran con el geoportal salesiano.			
Primary Key	id_per			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_per	serial	Yes	Identificación persona
FK	id_ciu	text	No	Código ciudad
	apellido_per	text	No	Apellido de la persona
	nombre_per	text	No	Nombre de la persona

	fechanac_per	date	No	Fecha de nacimiento de la persona
	correo_per	text	No	Correo electrónico de la persona
	telefono_per	text	No	Teléfono de la persona
	direccion_per	text	No	Dirección de la persona
	fax_per	text	No	Fax de la persona
	direcciontrab_per	text	No	Dirección trabajo de la persona
	fechacrea_per	date	No	Fecha creación del registro
	fechamodif_per	date	No	Fecha modificación persona
	estado_persona	boolean	No	Estado del registro

Nota. Diccionario de datos tabla persona

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 12. *Diccionario de datos tabla usuario*

Nombre	tb_usuario			
Descripción	Almacena la información de los usuarios registrados en el sistema.			
Primary Key	id_usu			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_usu	Serial	Yes	Identifica el usuario
FK	id_rol	Integer	Yes	Código del rol
FK	id_per	Integer	Yes	Código de la persona
	usuario_usu	Text	No	Usuario login
	contrasenia_usu	Text	No	Contraseña
	fechacrea_usu	Text	No	Fecha creación del registro
	fechamodif_usu	Text	No	Fecha modificación del registro
	fecmodifcontras_usu	Text	No	Fecha modificación contraseña
	estado_usu	Boolean	No	Estado del registro
	verifresetcontra_usu	Text	No	Verificación de la contraseña

Nota. Diccionario de datos tabla usuario

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 13. *Diccionario de datos tabla rol*

Nombre	tb_rol			
Descripción	Almacena la información de los roles que el usuario podrá desempeñar en el sistema.			
Primary Key	id_rol			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_rol	serial	Yes	Identifica el rol
	nombre_rol	text	No	Nombre del rol
	descripcion_roll	text	No	Descripción del rol
	estado_rol	boolean	No	Estado del registro

Nota. Diccionario de datos tabla rol

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 14. *Diccionario de datos tabla módulo rol*

Nombre	tb_modulorol			
Descripción	Tabla intermedia entre el módulo y el rol que rompe la relación varios a varios.			
Primary Key	id_modul, id_rol,			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_modul	serial	Yes	Identifica el módulo
PK	id_rol	serial	Yes	Identifica el rol
	id_menu	integer	Yes	Identifica el menú

Nota. Diccionario de datos tabla modulo rol

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 15. *Diccionario de datos tabla módulo*

Nombre	tb_modulo			
Descripción	Almacena la información de cada uno de los módulos que el geoportal tiene.			
Primary Key	id_modul			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_modul	serial	Yes	Identifica el módulo
	nombre_modul	text	No	Nombre del módulo
	imagen_modul	text	No	Ícono del módulo
	orden_modul	integer	No	Orden del módulo

	estado_modul	boolean	No	Estado del registro

Nota. Diccionario de datos tabla módulo

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 16. *Diccionario de datos tabla submódulo*

Nombre	tb_submodulo			
Descripción	Almacena la información de los submódulos que estarán ligados a un módulo.			
Primary Key	id_submod			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_submod	serial	Yes	Identifica el submódulo
FK	id_modul	integer	Yes	Código del módulo
	nombre_submod	text	No	Nombre del submódulo
	icono_submod	text	No	Icono del submódulo
	archivo_submod	text	No	Descripción del submódulo
	orden_submod	integer	No	Orden del submódulo
	estado_submod	boolean	No	Estado del archivo

Nota. Diccionario de datos tabla submódulo

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 17. *Diccionario de datos tabla permiso*

Nombre	tb_permiso			
Descripción	Almacena la información de los permisos que tendrá el usuario.			
Primary Key	id_permiso			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_permiso	Serial	Yes	Identifica el permiso
FK	id_rol	Integer	Yes	Código del rol
FK	id_submod	Integer	Yes	Código del submódulo
	ver_permiso	Boolean	No	Permiso para ver información
	editar_permiso	Boolean	No	Permiso para editar información
	eliminar_permiso	Boolean	No	Permiso para eliminar información
	generar_permiso	Boolean	No	Permiso para generar archivo

	importar_permiso	Boolean	No	Permiso para importar archivo
	exportar_permiso	Boolean	No	Permiso para exportar archivo
	estado_permiso	Boolean	No	Estado del archivo

Nota. Diccionario de datos tabla permiso

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 18. *Diccionario de datos tabla perfil submódulo*

Nombre	tb_perfil_tb_submodulo			
Descripción	Tabla intermedia entre tb_perfil y tb_submodulo			
Primary Key	id_rol, id_submod			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_rol	serial	Yes	Identifica el rol
PK	id_submod	serial	Yes	Identifica el submódulo

Nota. Diccionario de datos tabla perdil submódulo

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 19. *Diccionario de datos tabla herramientas*

Nombre	tb_herramientas			
Descripción	Almacena las funcionalidades que dispondrá el usuario.			
Primary Key	id_herramientas			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_herramientas	Serial	Yes	Identifica las herramientas
	nombre_herramienta	Text	No	Nombre de las herramientas

Nota. Diccionario de datos tabla herramientas

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 20. *Diccionario de datos tabla auditoria*

Nombre	tb_auditoria			
Descripción	Tabla de auditoría que se utiliza para registrar los cambios que realicen los usuarios en el módulo de administración.			
Primary Key	id_auditoria			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**

PK	id_auditoria	serial	Yes	Identifica el registro de auditoría
	usuario	Text	No	Nombre del usuario que realizo la acción
	submodulo	Text	No	Sobre que submódulo realizo la acción
	accion	Text	No	Nombre de la acción que realizo el usuario
	fecha	Timestamp	No	Fecha que realizo la acción en el smódulo.

Nota. Diccionario de datos tabla auditoría.

Elaborado por: Sandoval Byron &Tutillo Oswaldo

3.3 Diseño conceptual.

Para representar de una mejor manera las nuevas funcionalidades desarrolladas para la versión 2 del Módulo de administración, se establece un esquema conceptual que represente las clases y relaciones que esta funcionalidad tendrá.

Al ser un proyecto que utiliza JSF el diagrama de clases se tomara como entidades que interactúan para cada una de las nuevas funcionalidades creadas para esta versión.

Cabe destacar que para realizar estos diseños fueron consideradas únicamente las nuevas funcionalidades creadas para esta versión.

3.3.1 Diseño conceptual de la funcionalidad creación menú.

El siguiente diseño conceptual representa las clases y relaciones que interactúan para la creación y acceso del menú para el módulo de administración.

En la figura 6 podemos observar el diseño conceptual de la creación del menú y cada una de las clases que interactúan para esta funcionalidad.

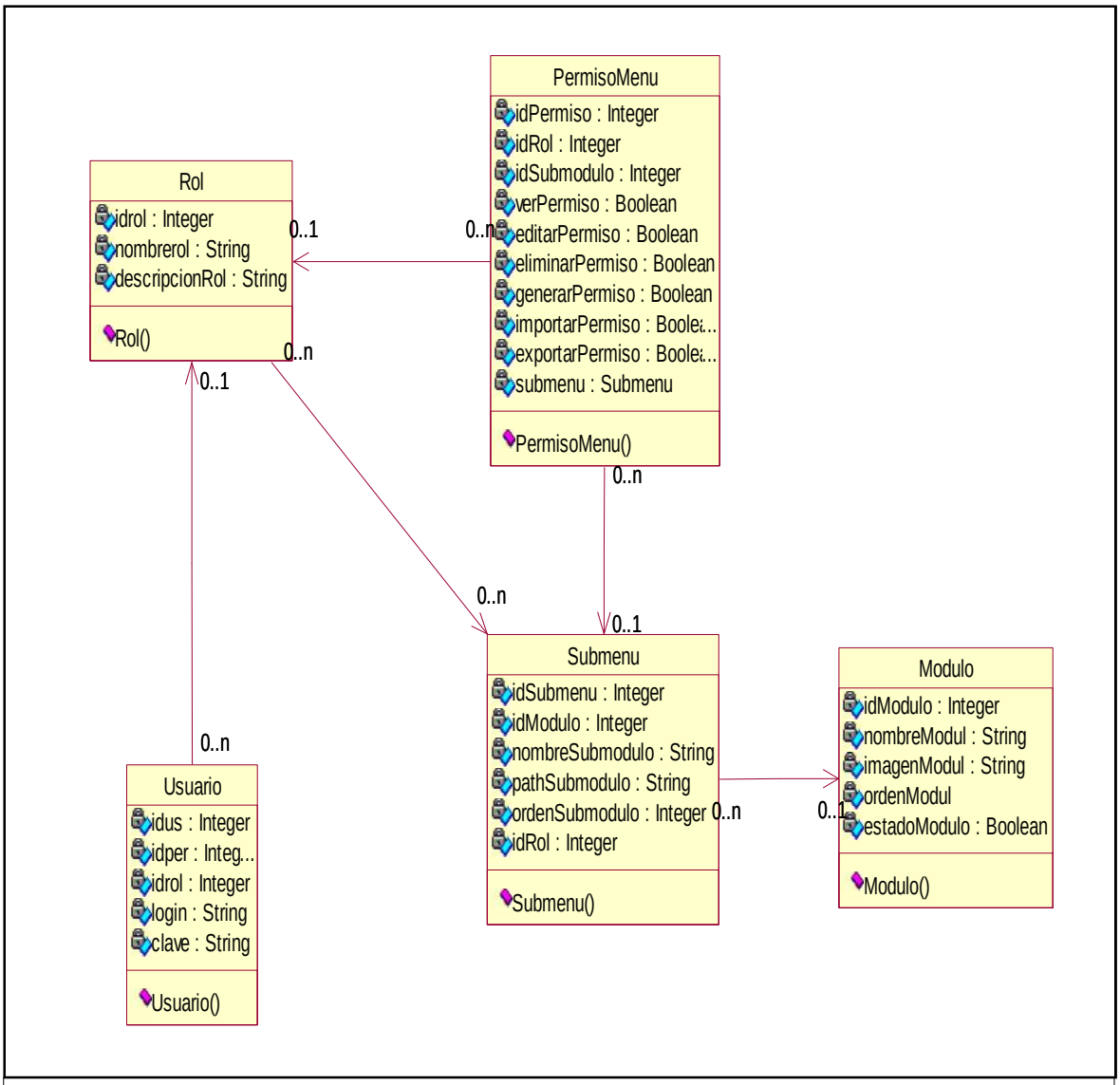


Figura 6. Diseño conceptual creación menú

Elaborado por: Sandoval Byron &Tutillo Oswaldo

3.3.2 Diseño conceptual de la funcionalidad seguridad de usuarios.

El siguiente diseño conceptual representa las clases y relaciones que interactúan para la autentificación del usuario cuando se ingresa al sistema.

En la figura 7 podemos observar el diseño conceptual del esquema de seguridad para el geoportal salesiano y cada una de las clases que interactúan para esta funcionalidad.

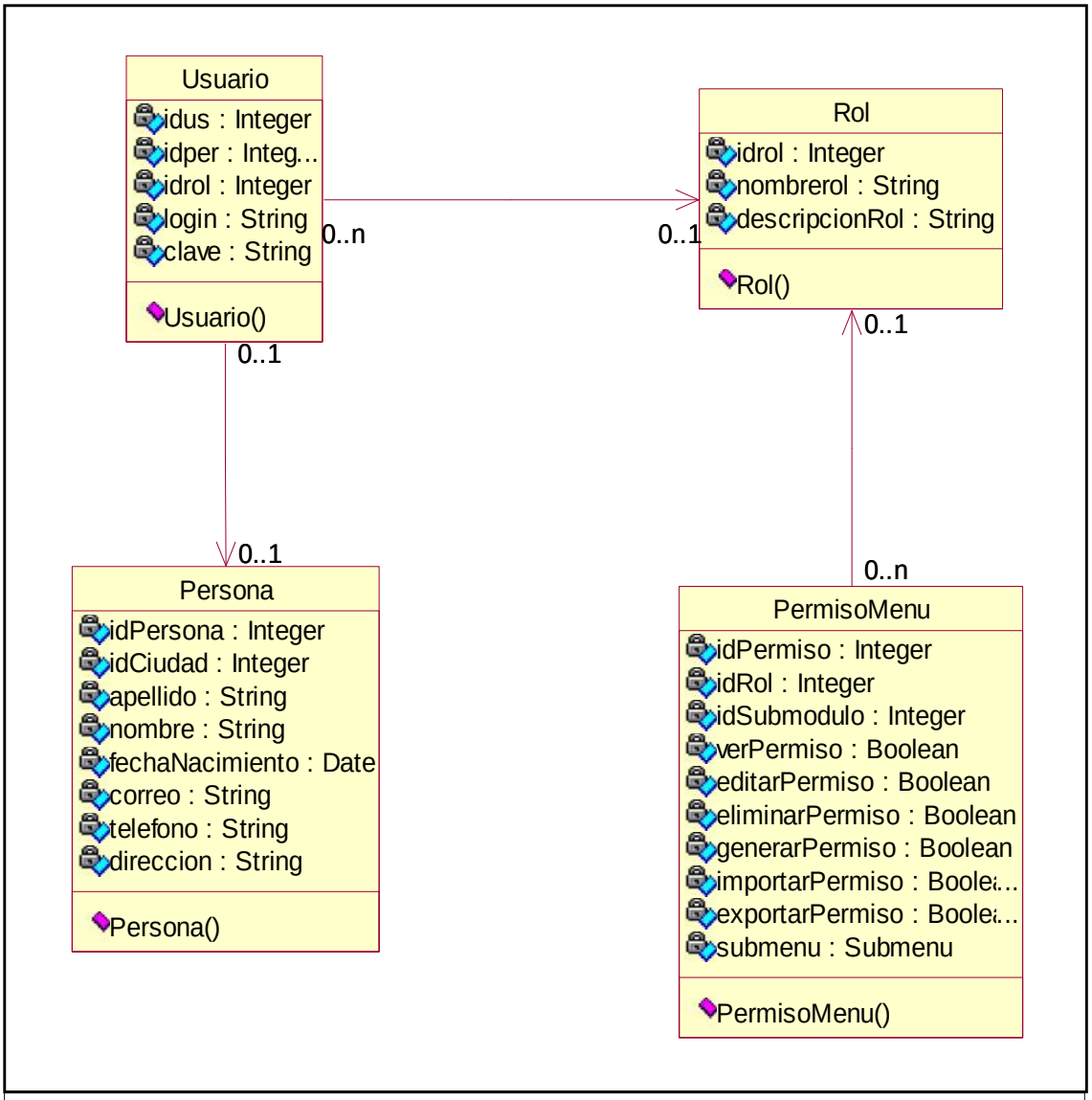


Figura 7. Diseño conceptual del esquema de seguridad de usuario

Elaborado por: Sandoval Byron &Tutillo Oswaldo

3.3.3Diseño conceptual de la funcionalidad creación y descarga de archivos GeoJSON.

El siguiente diseño conceptual representa las clases y relaciones que interactúan para la creación y descarga de un archivo en el formato .geojson.

Cabe mencionar que el resultado de este diseño se utiliza para la funcionalidad de la creación de una tabla en la base de datos, con el resultado de un query en el sub-modulo Generador de archivos geojson.

En la figura 8 podemos observar el diseño conceptual de la creación y descarga de archivos geojson y cada una de las clases que interactúan para esta funcionalidad.

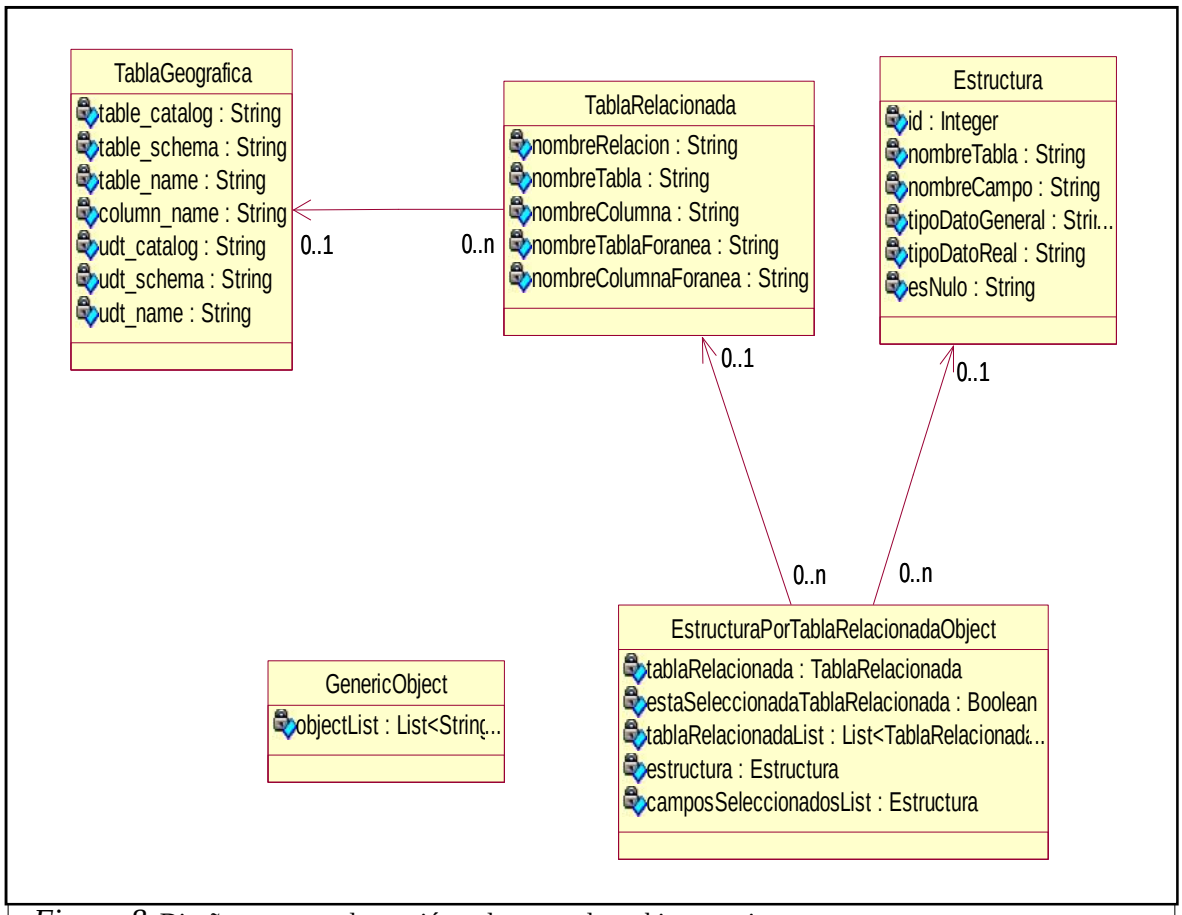


Figura 8. Diseño conceptual creación y descarga de archivos geojson

Elaborado por: Sandoval Byron &Tutillo Oswaldo

3.4 Diseño de interfaz abstracta.

Una ADV o (Vista de Datos Abstracta) por su siglas en ingles, es una interfaz representada por diferentes elementos con los que el usuario va a interactuar. La ADV contiene los elementos principales que se mostrarán al usuario cuando accede a una determinada sección. La apariencia real de los atributos y el diseño final de la ADV en la pantalla real se hace en la fase de la implementación.

A las ADV'S se las puede definir como un mapa de maquetación, la cual es la representación gráfica de cada una de las pantallas que tendrán esta versión.

3.4.1 Funcionalidad menú dinámico.

Se utiliza para definir la interfaz de usuario para el menú dinámico del módulo de Administración. El menú debe estar ubicado bajo la imagen del encabezado de la página, para que el usuario lo pueda identificar fácilmente. Esta interfaz está presente desde que el usuario ingresa a la pantalla principal de la aplicación web.

En la figura 9 se puede observar el ADV del menú inicial.

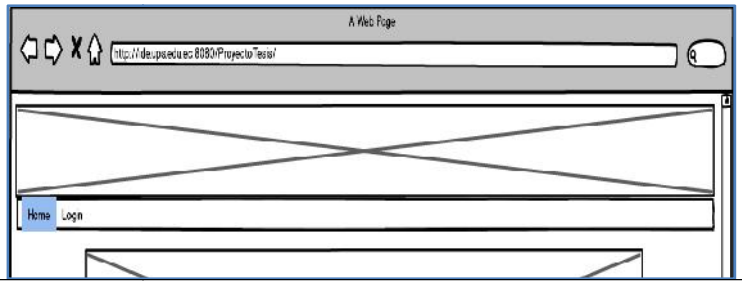


Figura 9. ADV's Menú inicial del módulo de Administración.
Elaborado por: Sandoval Byron &Tutillo Oswaldo.

Cuando el usuario hace clic en el menú *Login*, se muestra una ventana emergente para que se pueda ingresar el nombre de usuario y la contraseña.

En la figura 10 se puede observar el ADV de la pantalla emergente al momento de ingresar el usuario.

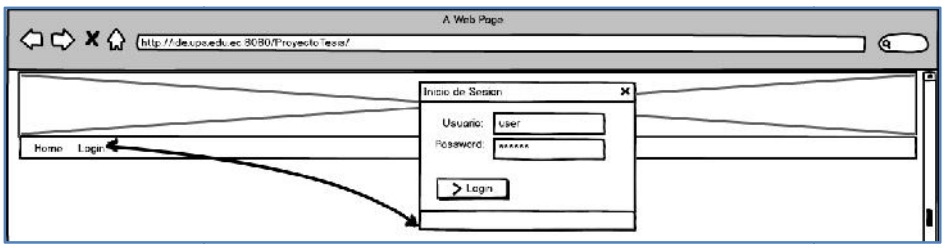


Figura 10. ADV's Pantalla emergente para ingreso de nombre de usuario y contraseña, para ingreso al módulo de Administración.
Elaborado por: Sandoval Byron &Tutillo Oswaldo.

Si el usuario ha ingresado correctamente los datos, se cargará la pantalla principal mostrando las opciones de menú del módulo de Administración a las que el usuario tiene permiso de acuerdo a su perfil. Además se mostrará una opción de menú indicando el usuario que ingreso al sistema con la opción para cerrar la sesión y el menú para regresar a la página principal del modulo de Administración.

En la figura 11 se muestra el ADV de la pantalla principal del módulo de Administración con el menú desplegado.

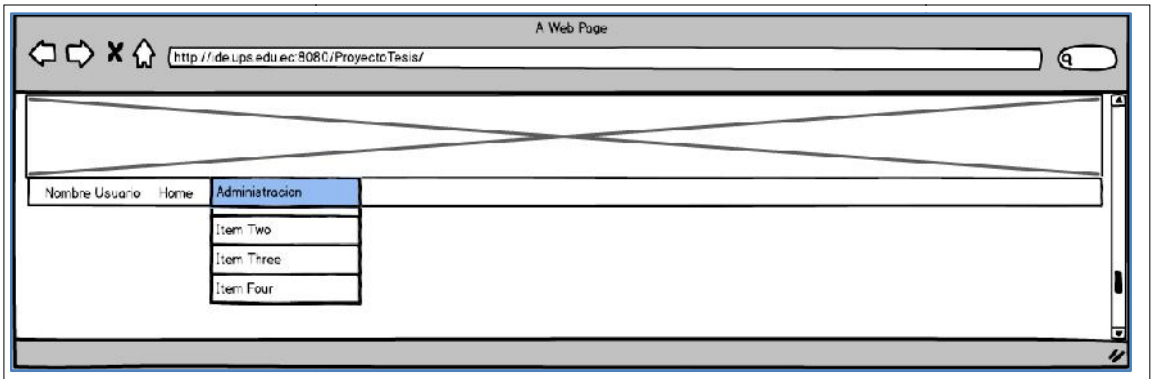


Figura 11. ADV's Pantalla principal del módulo de administración, con opciones de menú cargadas de acuerdo al usuario.
Elaborado por: Sandoval Byron &Tutillo Oswaldo.

3.4.2 Funcionalidad de generación de área de influencia.

Sirve para definir la interfaz de usuario de la funcionalidad *Generador Área de Influencia*. Esta funcionalidad, permite la georeferenciación del área de influencia de una Obra Salesiana. El polígono correspondiente al área de influencia, es trazado en un mapa que se muestra al inicio de la pantalla de la funcionalidad. Para la asignación del polígono que representa el área de influencia, a un registro de la base de datos; la interfaz de usuario muestra listas desplegables dependientes, que cargan la información en base a la selección del usuario. Las listas desplegables corresponden a información de Casa Salesiana, Obra Salesiana y Lugar.

En la figura 12, se observa el ADV con la funcionalidad de generación del área de influencia.

Figura 12. ADV's. Pantalla principal de la funcionalidad Generador Área de Influencia, para la georeferenciación de polígonos.

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

3.4.3 Funcionalidad de exportación de archivos Geojson.

Sirve para definir la interfaz de usuario de la funcionalidad para la exportación de archivos con formato GeoJSON (*.geojson). Para que un usuario pueda acceder a esta pantalla, se necesita que este registrado en el sistema y que cuente con los permisos correspondientes.

En la figura 13 podemos observar el ADV con la funcionalidad del exportador GeoJSON.

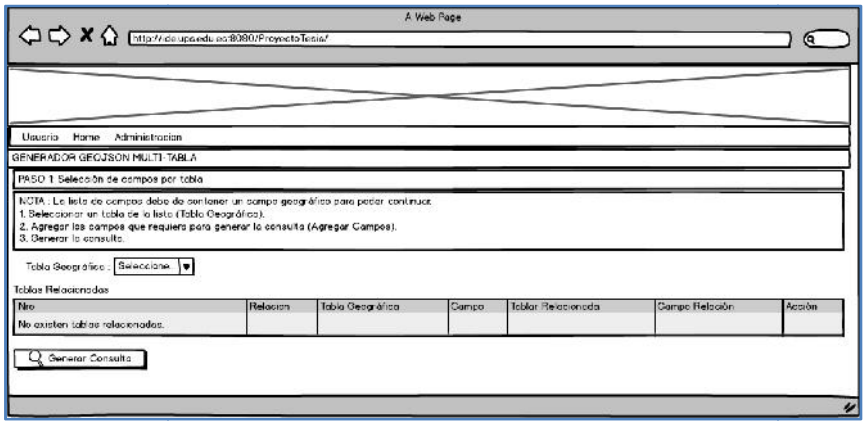


Figura 13. ADV's. Pantalla principal de la funcionalidad Exportador GeoJson

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

Los procesos para generar un archivo en formato GeoJSON, deben mostrarse divididos en secciones por cada uno de los pasos en el que se ha dividido el proceso principal. Se considera dos pasos para el proceso de generación de archivos geojson que son:

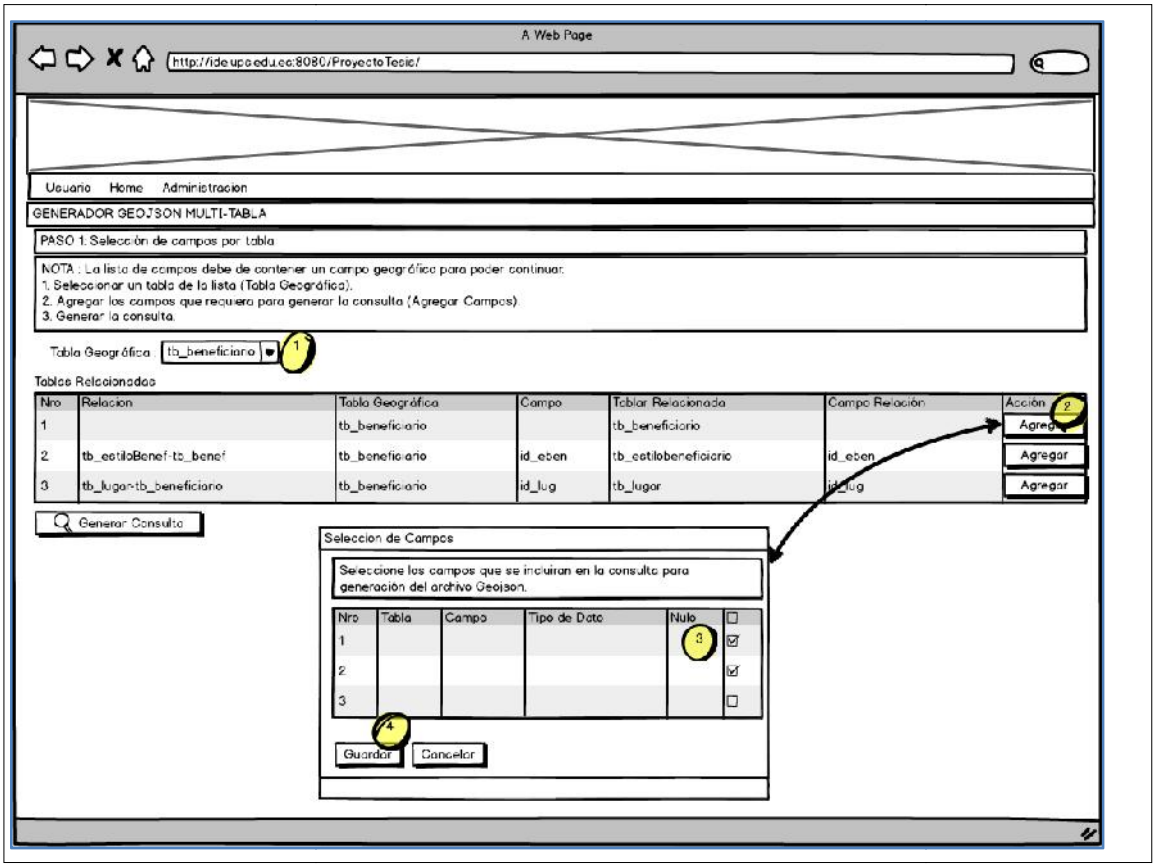
- Paso 1: Selección de campos por tabla.
- Paso 2: Generación de Archivo JSON.

Cuando un usuario ingresa a la funcionalidad, se muestra únicamente los campos relacionados al paso número 1 que consta de: leyendas con texto informativo, lista desplegable con los nombres de las tablas de la base de datos que contengan campos de tipos de dato *geometry*, tablas relacionadas sin registros y el botón Generar Consulta deshabilitado.

- **Paso 1: Selección de campos por tabla.**

  En esta sección, se considera el diseño de interfaces de usuario que permiten la selección de información necesaria para la generación de archivos.

  Cuando el usuario selecciona un ítem de la lista desplegable, se muestra la información relacionada en la tabla resumen. Además se habilita la ventana emergente para que el usuario seleccione los campos de una tabla, por medio del botón Agregar Campos. Esto se puede observar en la figura 14.



Nota. Cuando el usuario hace clic en el botón *Agregar Campos*, se debe desplegar una pantalla emergente con una tabla resumen con los campos que contiene la tabla de base de datos seleccionada. La tabla resumen de la ventana emergente debe disponer de una columna que permita seleccionar uno o varios campos al usuario.

Pasos:

1. Escoger la tabla que contiene datos geográficos.
2. Seleccionar la tabla de la cual se escogerán los campos para la generación del archivo
3. Escoger los campos que formaran parte del archivo
4. Guardar la selección

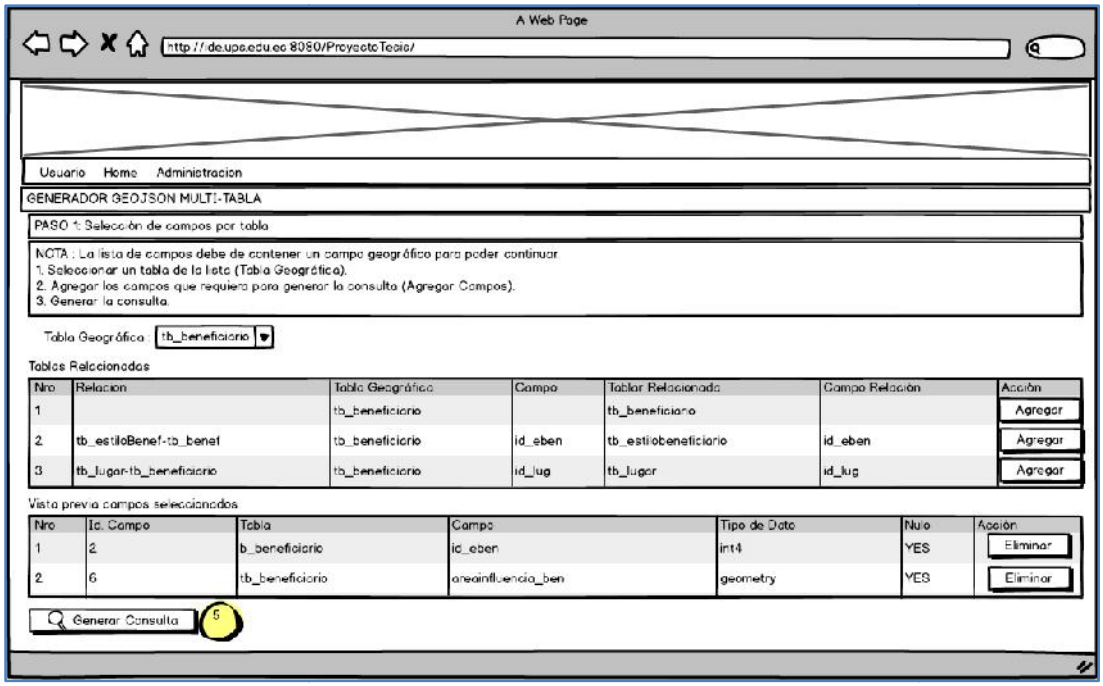
Figura 14. ADV's. Funcionalidad Exportador GeoJson, que muestra la presentación del detalle de una tabla seleccionada y la ventana emergente para la selección de campos.

Elaborado por: Sandoval Byron &Tutillo Oswaldo

La funcionalidad para poder generar una consulta SQL, a través de la información seleccionada por el usuario, está disponible a través del botón *Generar Consulta*; que se habilita cuando el usuario completa la información previa. La información, a ser incluida

en la consulta SQL (campos seleccionados), debe mostrarse en una tabla común, cuando sea de diferentes tablas.

En la figura 15 podemos observar el ADV con la funcionalidad del exportador GeoJSON seleccionando las columnas para generar la consulta.



Pasos:

5. Se presiona sobre el botón generar consulta para que se realice el query de acuerdo a los aspectos seleccionados.

Figura 15. ADV's Funcionalidad Exportador GeoJson, que muestra la presentación los campos seleccionados por el usuario para ser incluidos en la consulta SQL en la sección Vista previa campos seleccionados.

Elaborado por: Sandoval Byron &Tutillo Oswaldo

- **Pasó 2: Generación de archivo GeoJSON.**

  En el paso número 2 para la generación del archivo GeoJson, se contempla que la interfaz de usuario disponga de una sección para poder visualizar la consulta SQL; generada a partir de la información seleccionada en él, la sección 1 y una tabla de resumen con la información resultante al ejecutar la consulta SQL. Las

columnas de la tabla resumen, son mostradas dinámicamente, dependiendo de los campos seleccionados por el usuario en la sección numero 1.

En la figura 16 podemos observar el ADV con la funcionalidad del exportador GeoJSON, mostrando el resultado de la consulta.

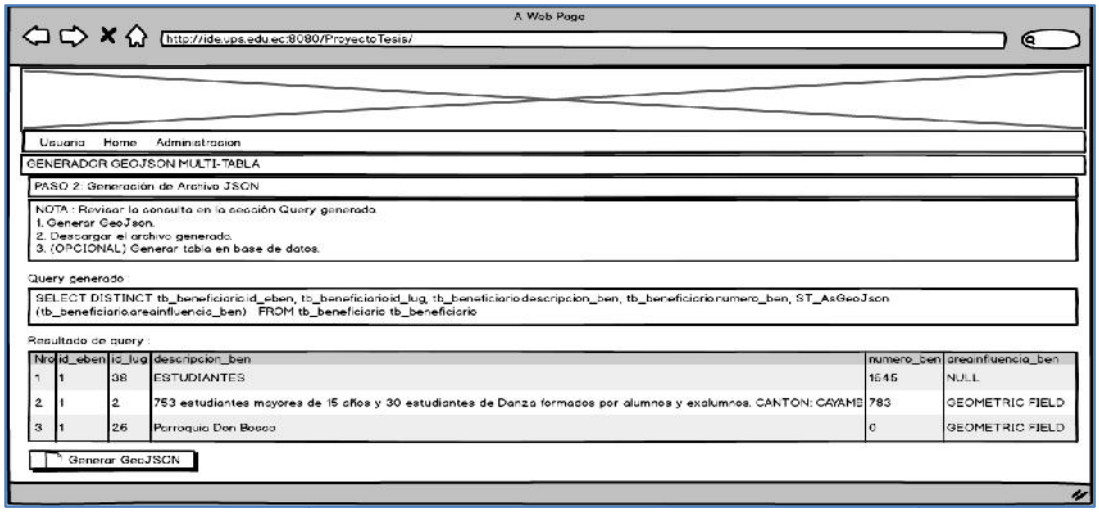
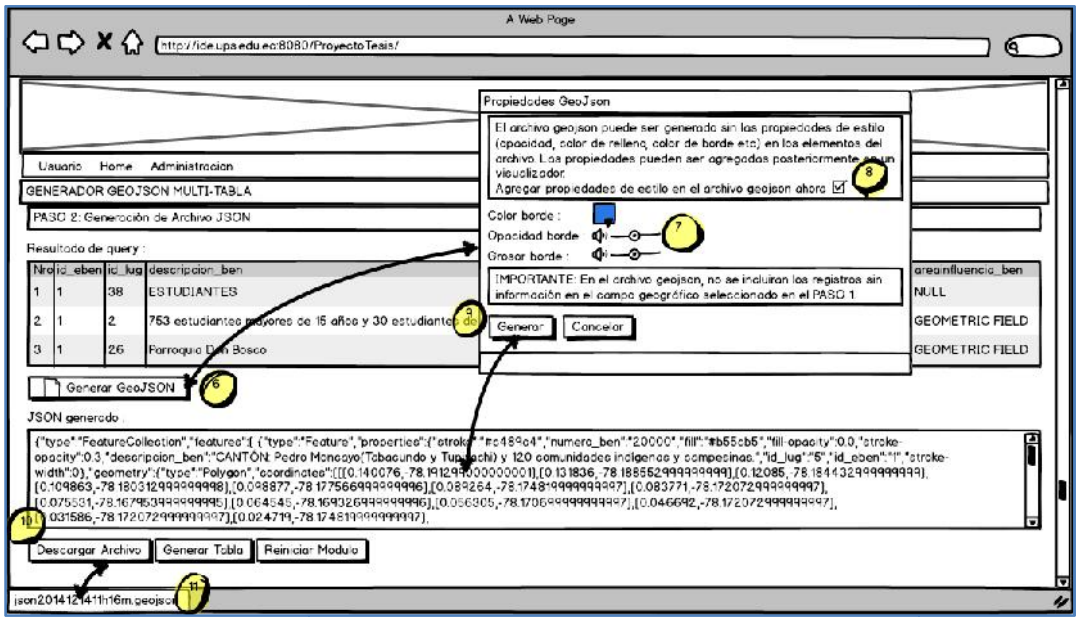


Figura 16. ADV's Funcionalidad Exportador GeoJson, que muestra el texto de la consulta SQL creada y los datos resultantes al ejecutar la misma consulta.
Elaborado por: Sandoval Byron &Tutillo Oswaldo.

Para poder continuar con la generación del archivo, el usuario debe hacer clic en el botón *Generar GeoJSON*. Al hacer clic en el botón *Generar GeoJSON*, se habilitará la ventana emergente *Propiedades GeoJson*; para la selección de parámetros de estilo como: opacidad, grosor; dependiendo del tipo de geometría del campo geográfico seleccionado en el paso número 1 e incluido en la consulta SQL.

Cuando el usuario, hace clic en el botón *Generar*, del la ventana emergente se habilita en la pantalla principal de la funcionalidad *Exportador GeoJson*, la sección *JSON generado*; con el texto correspondiente al contenido del archivo geojson y los botones *Descargar Archivo*, *Generar Tabla* y *Reiniciar Módulo*. En la figura 17 podemos observar el ADV con la funcionalidad del exportador GeoJSON, con la opción para descargar el archivo.



Nota: La selección de las propiedades de estilo de la ventana emergente *Propiedades GeoJson*, son opcionales y el usuario podrá elegir si llenarlas o no. La interfaz de usuario debe mostrar los mensajes de advertencia correspondientes.

Pasos:

6. Presionar sobre el botón generar GeoJSON en cual abrirá un popup.
7. Escoger la opacidad del borde de la traza el grosor y el color del mismo.
8. Señalar si se desea agregar propiedades de estilo a la traza.
9. Presionar el botón generar para que agregue los aspectos visuales escogidos al archivo GeoJSON.
10. Presionar el botón Descargar archivo para que descargue el archivo en formato GeoJSON.
11. Dependiendo del navegador, el archivo GeoJSON se podrá observar en la parte inferior del navegador.

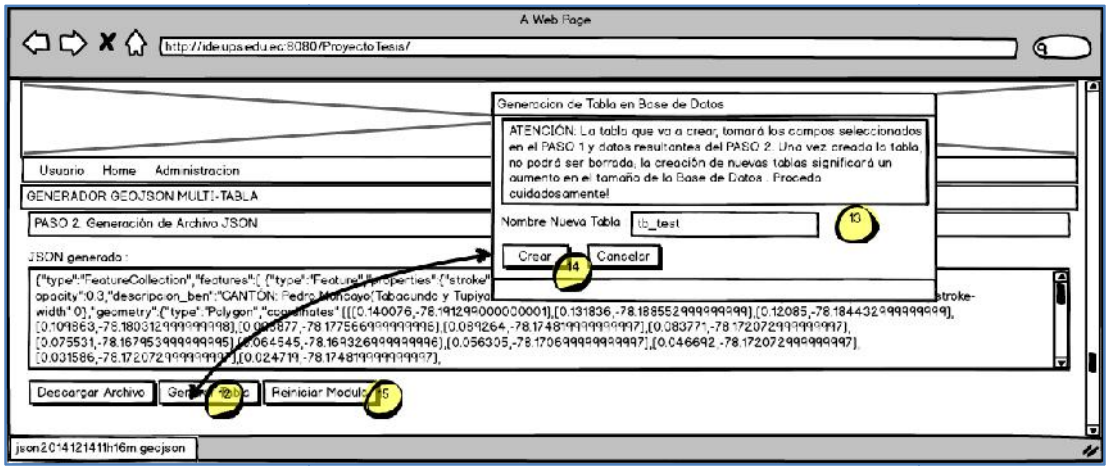
Figura 17. ADV's Funcionalidad Exportador GeoJson, que muestra la generación del archivo geojson.
Elaborado por: Sandoval Byron &Tutillo Oswaldo.

3.4.4 Funcionalidad generar tabla.

Se utiliza para definir la interfaz de usuario que permita la creación de una nueva tabla en la base de datos. La interfaz para esta funcionalidad debe de ser incluida dentro de la interfaz de usuario del Exportador de archivos GeoJson, como una ventana emergente a través del botón *Generar Tabla*. Se incluye esta funcionalidad dentro del Exportador de archivos GeoJson, ya que la nueva tabla que se cree en la base de datos debe de tomar la estructura seleccionada por el usuario y los datos resultantes en esta funcionalidad. Esta

funcionalidad permite al usuario ingresar únicamente el nombre de la nueva tabla, para que sea creada en la base de datos. El nombre de la tabla será ingresado en una caja de texto que se mostrara en la ventana emergente, debajo del mensaje de advertencia. También se incluye los botones *Crear* y *Cancelar*.

En la figura 18 podemos observar el ADV con la funcionalidad del exportador GeoJSON, con los pasos desde el 12 hasta el paso 15.



Nota: El uso de esta funcionalidad es opcional para el usuario y la interfaz debe de mostrar los respectivos mensajes de advertencia.

Pasos:

12. Al presionar el botón Generar tabla se abrirá un popup.
13. Se escribirá el nombre de la nueva tabla a crearse.
14. Al presionar el botón crear se creara la tabla con el nombre descrito en el paso anterior.
15. Al presionar el botón Reiniciar módulo permitirá reiniciar todo el proceso de creación del archivo para generar otro con otras selecciones.

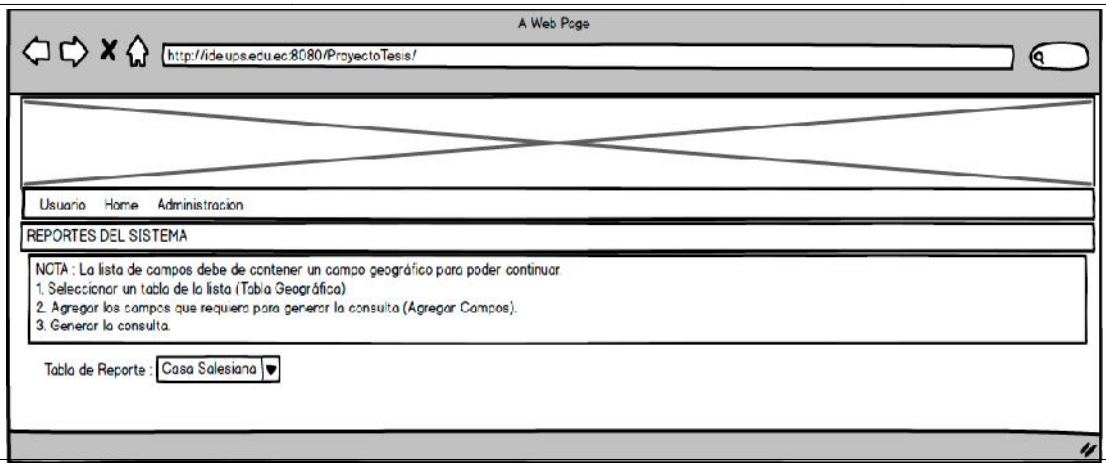
Figura 18. ADV's Funcionalidad Generar Tabla, que muestra la ventana emergente para la creación de una nueva tabla en la base de datos.

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

3.4.5 Funcionalidad generar reportes.

El diseño de esta interfaz, se usara para la generación de reportes para el usuario. Para la selección del tipo de reporte, el usuario dispone de un campo de selección en donde se listan los reportes disponibles en el sistema.

En la figura 19 podemos observar el ADV con la funcionalidad de la generación de reportes.



Nota: La interfaz de usuario, dispone de un área de texto donde se muestra las instrucciones para generar un reporte y un botón que permita generar y descargar el reporte seleccionado al final de la interfaz.

Figura *19.* ADV's Funcionalidad Generar Reportes, para generar reportes disponibles en el sistema.
Elaborado por: Sandoval Byron &Tutillo Oswaldo.

Cuando el usuario selecciona el tipo de reporte: Obra Salesiana, se muestra la interfaz de usuario; en donde se muestran campos de selección desplegables para seleccionar la Casa y Obra Salesiana. También, el detalle de la información seleccionada es mostrado en tablas.

En la figura 20 podemos observar el ADV con la funcionalidad de la generación de reportes de una obra Salesiana.

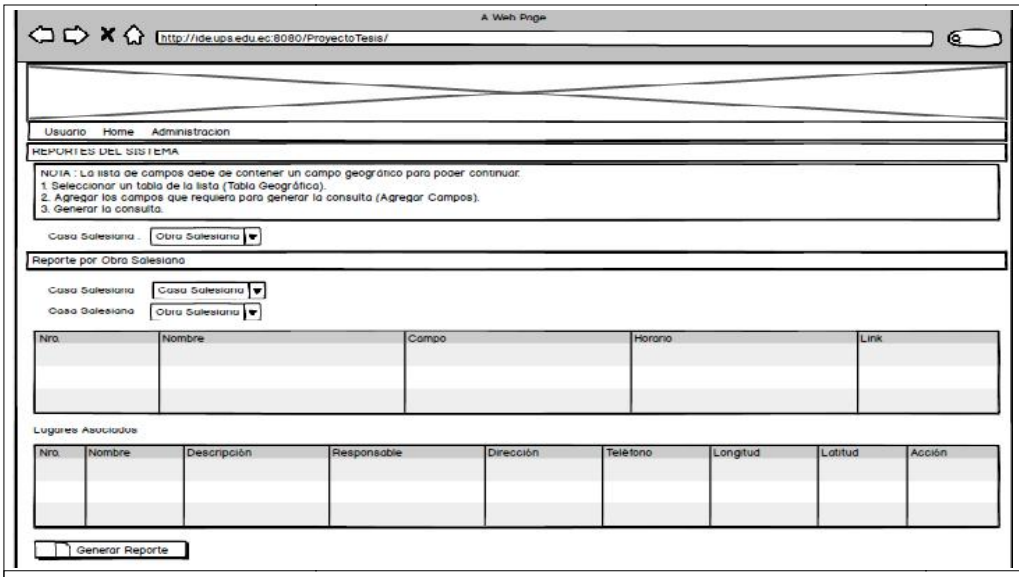


Figura 20. ADV's Funcionalidad Generar Reportes, para generar un reporte por Obra Salesiana.

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

Para el reporte de tipo: Casa Salesiana, se muestra la interfaz de usuario en donde la información es seleccionada de un campo de selección desplegable. También, el detalle de la información seleccionada se muestra en una tabla.

En la figura 21 se observa el ADV que representa la funcionalidad para generar el reporte por cada una de las casas salesianas.

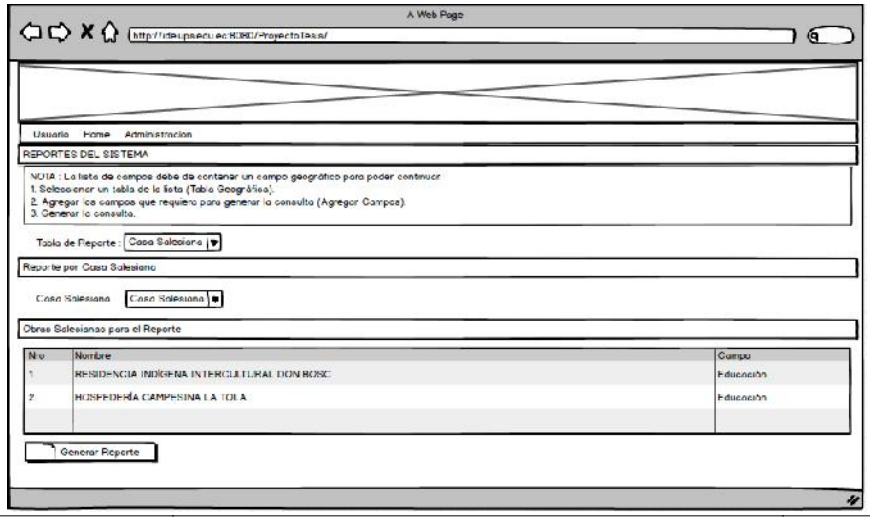


Figura 21. ADV's Funcionalidad Generar Reportes, para generar un reporte por Casa Salesiana.

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

3.4.6 Auditoria de usuarios (Bitácora).

El diseño de esta interfaz, se usara para el monitoreo de las acciones, que realice el usuario en el sistema, cabe indicar que por ahora esta opción solo estará disponible en el perfil de administrador.

En la figura 22 se observa el ADV que representa la opción de auditoría para el módulo.

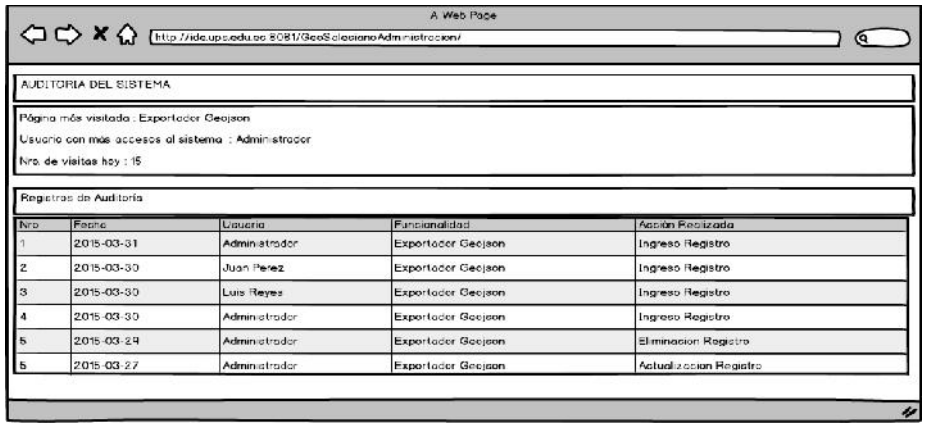


Figura 22. ADV's. Auditoria del módulo.

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

3.5 Diagrama de clases.

El diagrama de clases que se muestra en la figura 22 siguiente, muestra las entidades, controladores y servicios de las nuevas funcionalidades que serán implementadas en la nueva versión del módulo de Administración del Geoportal Salesiano.

Cabe destacar que para la realización de este diagrama fueron considerados únicamente las clases que se implementaron para la realización de la nueva versión; únicamente las clases ***Persona***, ***ControladorPersona*** y ***ServicioPersona*** fueron modificadas de la anterior versión y se consideran como la unión con la anterior versión del módulo de administración.

Para observar el diagrama de clases del módulo de administración en su versión 1, se lo puede revisaren el **Anexo 6. Diagrama de clases versión 1.**

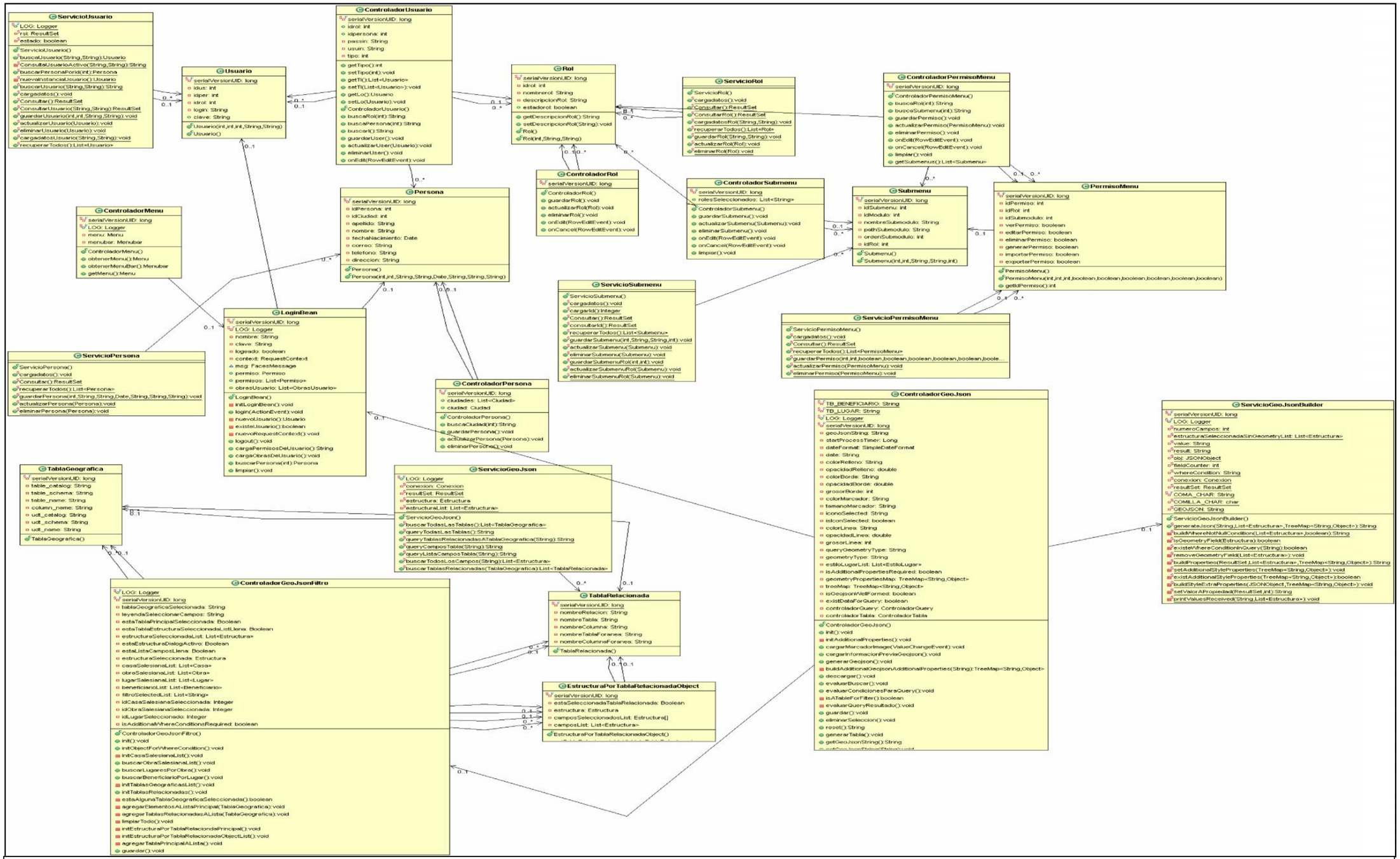


Figura 23. Diagrama de clases

Elaborado por: Sandoval Byron & Tutillo Oswaldo

3.6 Diagrama de componentes.

En el Geoportal Salesiano tanto en el módulo de Visualización como en el Gestor de datos geográficos en la versión 1; está construido con ayuda de las herramientas libres, el diagrama de componentes muestra los componentes físicos del sistema y las relaciones entre los componentes (elementos de software) que proporcionan o consumen mediante las interfaces.

En vista que el presente trabajo representa la versión 2 del módulo de administración, es necesario que siga utilizando los mismos componentes que en la versión anterior.

Es por ello que para esta sección se utilizo el diagrama y la descripción desarrollada en el trabajo de grado intitulado análisis, diseño e implementación del módulo de visualización y gestión de datos geográficos para el geoportal de la comunidad salesiana versión 2.0 desarrollado por los señores Cofre Víctor y Toledo Stalin; con ello se especifica que el Geoportal Salesiano, sigue los mismos lineamientos y utiliza los mismos componentes que en sus anteriores versiones.

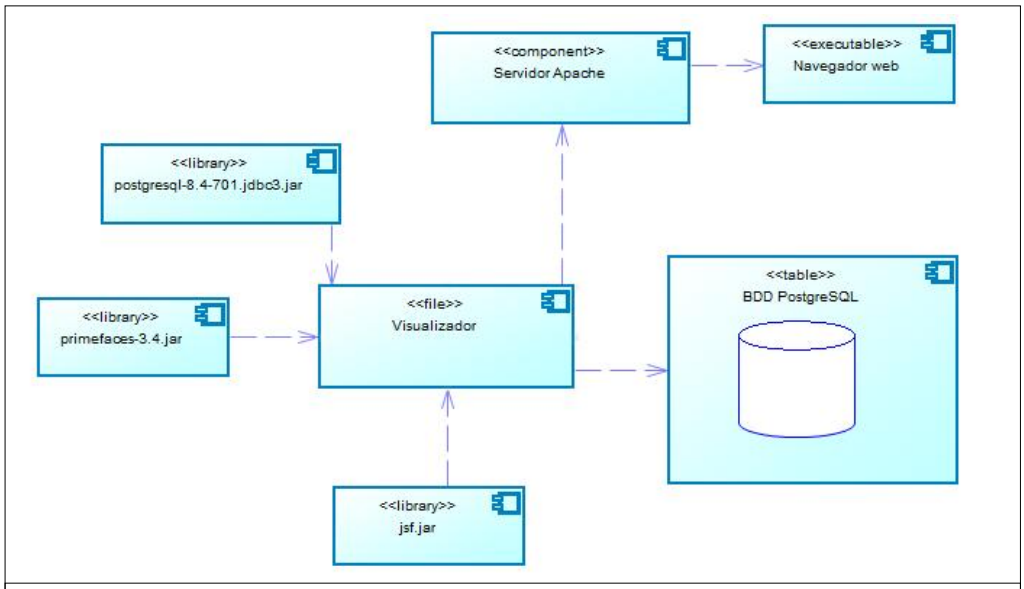


Figura 24. Diagrama de componentes del módulo de visualización.

Elaborado por: Cofre Víctor & Toledo Stalin

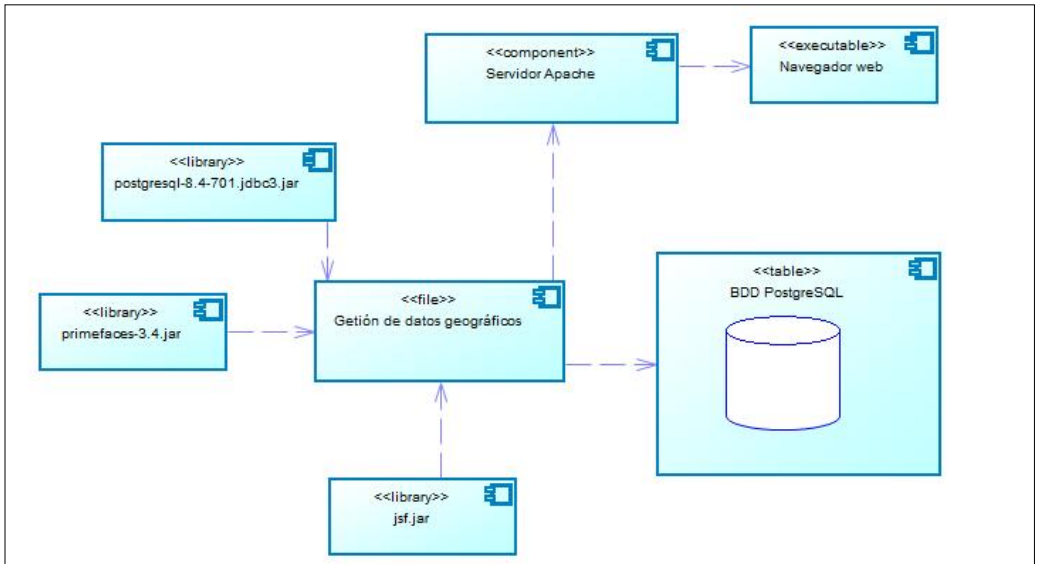


Figura 25. Diagrama de componentes del módulo de gestión de datos geográficos
Elaborado por: Cofre Víctor & Toledo Stalin

Descripción de los componentes de software:

- **Servidor Apache:** Es un servidor web HTTP de código abierto, multiplataforma que se utiliza para alojar la presente aplicación y que pueda ser utilizada a través de internet.
- **Java:** Lenguaje de programación que trabaja con librerías como primefaces, postgres, jsf, etc. que se utiliza para el desarrollo de la aplicación.
- **PostgreSQL:** Almacena la información que la aplicación consumirá para mostrar al usuario la información requerida.
- **Primefaces:** Es una librería que contiene muchos componentes que ayudan en el diseño de una interfaz agradable al usuario y que facilita el desarrollo de aplicaciones web.
- **JSF (Java Server Faces):** Es una tecnología y framework para aplicaciones Java que facilita el desarrollo de interfaces de usuario ya que dispone de una gran cantidad de componentes.

(Stalin, 2014).

CAPÍTULO4
DESARROLLO

En el presente capítulo, se describe el código desarrollado para cada funcionalidad incorporada en el módulo de Administración del geoportal de la Comunidad Salesiana. Se expondrá los fragmentos del código más importantes de cada funcionalidad, que serán tomados de las diferentes capas en las cuales está dividido módulo de Administración.

4.1 Framework de seguridad.

Esta funcionalidad esta implementada en la clase ***LoginFilter***. Esta clase es un servlet que cumple la tarea de filtro web de seguridad, en donde se implementa la funcionalidad para filtrar los formularios que conforman el sistema, así como el acceso a los diferentes recursos disponibles. De la misma manera, este servlet implementa la funcionalidad para especificar los recursos que no son necesarios que sean protegidos en la aplicación.

Para el control de los recursos del sistema se utiliza el método ***doFilter***, del filtro de seguridad. Este método se encarga de filtrar los recursos de la aplicación, para lo cual el método ***doFilter*** realiza lo siguiente:

- Obtiene el bean ***LoginBean***, en donde se almacena la información del usuario.
- Procesa la url que está siendo requerida por el usuario. En proceso se pueden presentar los siguientes escenarios:

    - Si el recurso al que se trata de acceder está en la lista de recursos que no requieren protección, el filtro continúa normalmente y muestra en pantalla el recurso solicitado.
    - Si el usuario ha iniciado la sesión en el sistema, el filtro muestra al usuario los recursos a los que tiene acceso.
    - Si el usuario no ha iniciado la sesión y trata de ingresar a una funcionalidad digitando la url; el filtro re-direcciona

automáticamente al usuario a la página principal denominada index.jsf.

En el método 1, se especifica cómo se encuentra constituido el método **doFilter** y cuáles son los atributos que este método necesita.

Método 1.doFilter

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
* @fecha_creacion Abr 11, 2014
* @fecha_ultima_actualizacion Nov 10, 2014
* @refrencia  http://www.dataprix.com/blog-it/tendencias-tecnologicas/software-libre/login-control-acceso-basico-primefaces-paso-paso
 */

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException,
ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse res = (HttpServletResponse) response;
        LoginBean loginBean = (LoginBean) req.getSession().getAttribute("loginBean");

        String urlStr = req.getRequestURL().toString().toLowerCase();
        boolean noProteger = noProteger(urlStr);
        LOG.info(urlStr + " - desprotegido=[" + noProteger + "]");

        if (noProteger(urlStr)) {
                chain.doFilter(request, response);
                return;
        }

        if (loginBean == null || !loginBean.estaLogeado()) {
                res.sendRedirect(req.getContextPath() + "/index.jsf");
                return;
        }

        chain.doFilter(request, response);
}
```

De la misma manera, en el método ***noProteger*** están agregados los recursos para los que el filtro de seguridad no va a restringir el acceso. Solamente se incluyen los recursos necesarios para la página inicial del sistema que son:

- Archivo index.jsf
- Archivos de plantillas de diseño

- Archivos con diseño de encabezado de página.
- Archivos con diseño de pie de página.
- Imágenes de la cabecera de la página.
- Imágenes del pie de página.
- Imágenes de presentación.

Para los recursos que no están en la lista de recursos no protegidos, el método evalúa el valor verdadero o falso a devolver; validando el inicio de sesión o no en el sistema.

En el método 2, se especifica el método ***noProteger***; en el cual podemos observar como este método evalúa los nombres de las imágenes que son necesarios para formar la primera pantalla y además si la sesión se encuentra iniciada o no.

Método 2. noProteger

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion Abr 11, 2014
* @fecha_ultima_actualizacion Nov 10, 2014
* @refrencia  http://www.dataprix.com/blog-it/tendencias-tecnologicas/software-libre/login-control-acceso-basico-primefaces-paso-paso
 */

private boolean noProteger(String urlStr) {

        if (urlStr.endsWith("index.jsf")
        || (urlStr.endsWith("templateprincipal1.jsf")
        || (urlStr.endsWith("header1.jsf")
        || (urlStr.endsWith("template.jsf")
        || (urlStr.endsWith("header.jsf")
        || (urlStr.endsWith("piepaginatesis.png")
        || (urlStr.endsWith("ima1.jpg")
        || (urlStr.endsWith("ima2.jpg")
        || (urlStr.endsWith("ima3.jpg")
        || (urlStr.endsWith("ima4.jpg")
        || (urlStr.endsWith("bannertesis.png")
        || (urlStr.endsWith("fondo.jpg")
        || (urlStr.endsWith("ajax-loader.gif"))))))))))))))

        return true;
```

```
        if (urlStr.indexOf("/javax.faces.resource/") != -1)
                return true;


                return false;
}
```

4.2. Gestión de usuarios y de perfiles.

Con la finalidad de gestionar los distintos ámbitos necesarios para la administración del portal, se creó algunos formularios, para el ingreso y actualización de información para el Geoportal.

En vista que los formularios tienen la misma funcionalidad, han sido creados siguiendo un mismo estándar diferenciados únicamente por las tablas en las cuales se va a guardar la información.

Siendo así, se detalla a continuación los métodos más relevantes para realizar el formulario de gestión de usuarios y con ello servirá como ejemplo para los demás formularios que fueron desarrollados para este módulo.

A continuaciónen el método 3, mostramos un fragmento de código en el cual se muestra el método ***guardarUser***.

Método 3. guardarUser

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion Abr 15, 2014
* @fecha_ultima_actualizacion Nov 20, 2014
 */

public void guardarUser() throws Exception{


        ServicioUsuario.guardarUsuario(lo.getIdrol(), lo.getIdper(), lo.getLogin(),lo.getClave());

        FacesMessage msg = new FacesMessage("Usuario Guardado", lo.getLogin()+"");
        FacesContext.getCurrentInstance().addMessage(null, msg);
        limpiar();

}
```

Este método se utiliza para guardar la información enviada mediante el formulario de administración de usuario.

Este método básicamente utiliza la función que procede del servicio del usuario que guarda la información en la tabla de datos *tb_usuario*.

El método ***guardarUsuario***, inserta mediante la sentencia sql el registro en la tabla *tb_usuario*.

A continuación en el método 4, se muestra un fragmento del código del método ***guardarUsuario***.

Método 4. guardarUsuario

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion Abr 15, 2014
* @fecha_ultima_actualizacion Nov 29, 2014
 */

public static void guardarUsuario(int idrol, int idpersona, String usuario, String clave) throws Exception {
        cargadatos();
        clave = PasswordMD.getStringMessageDigest(clave, PasswordMD.PUBLIC_MD5);
        String sentencia = "insert into tb_usuario (id_rol,id_per,usuario_usu,contrasenia_usu,estado_usu) values ("
        + idrol + "," + idpersona + ",'" + usuario + "','" + clave + "',true);";

        Conexion con = new Conexion();
        String mensaje = con.Ejecutar(sentencia);

}
```

Cabe destacar que el atributo ***clave*** se encuentra encriptado mediante el método estático ***getStringMessageDigest*** el cual debe ser seleccionado que tipo de algoritmo de encriptado se va a escoger para realizar la encriptación mediante el atributo estático ***PasswordMD.PUBLIC_MD5***.

A continuación se muestra el método 5, el que se denomina ***getStringMessageDigest***.

Método5.getStringMessageDigest

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion Abr 30, 2014
* @fecha_ultima_actualizacionDic 01, 2014
 */

public static String getStringMessageDigest(String message, String algorithm) throws NoSuchAlgorithmException {
        byte[] digest = null;
        byte[] buffer = message.getBytes();
        try {
                MessageDigest messageDigest = MessageDigest.getInstance(algorithm);
                messageDigest.reset();
```

```
                messageDigest.update(buffer);
                digest = messageDigest.digest();
        } catch (NoSuchAlgorithmException e) {
                throw e;
        }
        return toHexadecimal(digest);

}
```

4.3 Gestión de acceso a usuarios.

Para poder acceder a las funcionalidades del sistema, un usuario debe de autenticarse en el sistema por medio de un nombre de usuario y una contraseña. De la misma manera, se cuenta con una funcionalidad que permite cerrar la sesión iniciada. La implementación de estas funcionalidades, se encuentran en la clase ***LoginBean***.

Para el inicio de sesión de un usuario en el sistema, se utiliza el método ***login***. Este método, contiene las instrucciones de código para evaluar los datos ingresados por el usuario; en la interface web. Así el método login, sigue el siguiente proceso:

- Verifica que el usuario exista en la base de datos del sistema. Si el usuario existe, se realizan las siguientes acciones:
  - Busca los permisos asignados al usuario, para las diferentes funcionalidades del sistema.
  - Busca la información adicional del usuario.
  - Notifica al filtro de seguridad del sistema, que el usuario si existe y que ha iniciado la sesión; por medio del atributo de nombre logeado que es de tipo boolean.
- Si los datos del usuario son incorrectos:
  - Notifica al filtro de seguridad del sistema que los datos del usuario son incorrectos.
  - Agrega la notificación para el usuario.

A continuación podemos observar el método 6 que se denomina ***login***, el cual se utiliza para permitir el acceso o no del usuario que se identifique, de acuerdo a las especificaciones indicadas anteriormente.

Método6.Login

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion May 01, 2014
* @fecha_ultima_actualizacionDic 12, 2014
 */

public void login(ActionEvent actionEvent) {
        LOG.info("****Inside Login Method in Login Bean!!!");
        nuevoRequestContext();
        try {
                if (existeUsuario()) {
                        logeado = true;
                        cargaPermisosDeUsuario();
                        buscarPersona(usuario.getIdper());
                        msg = new FacesMessage(FacesMessage.SEVERITY_INFO,
                        "Bienvenido : " + getPersona().getNombre() + " "+ getPersona().getApellido(), null);
                } else {
                        logeado = false;
                        msg = new FacesMessage(FacesMessage.SEVERITY_WARN, "Credenciales incorrectas", "");
                }

                FacesContext.getCurrentInstance().addMessage(null, msg);
                context.addCallbackParam("estaLogeado", logeado);
                if (logeado)
                context.addCallbackParam("view", "administracion/HomeDatos.jsf");
        } catch (Exception e) {
                LOG.error("###ERROR EN LOGIN" + e.getLocalizedMessage());
                FacesMessageUtil.addErrorMessage(Mensaje.TRANSACCION_FALLIDA);
        }
}
```

Por otra parte, para cerrar la sesión del usuario, se lo realiza a través del método ***logout***; el cual dispone de las instrucciones de código necesarias en donde se realiza las siguientes operaciones:

- Invalida los objetos de sesión, accedidos a través de la clase ***ExternalContext***.
- Re-direcciona el sistema hacia la página de inicio, a través del método ***redirect***.

Como se menciono anteriormente el método ***logout*** es utilizado por el ítem menú que fue creado para cerrar la sesión, a continuación en el método 7 se presenta un fragmento de este método con el cual podemos cerrar la sesión.

Método7. logout

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion May 06, 2014
* @fecha_ultima_actualizacionDic 12, 2014
 */

public void logout() throws IOException {
        LOG.info("*****Dentro de logout");

        FacesContext context = FacesContext.getCurrentInstance();
        ExternalContext externalContext = context.getExternalContext();

        //invalidate session
        Object session = externalContext.getSession(false);
        HttpSession httpSession = (HttpSession) session;
        httpSession.invalidate();

        msg = new FacesMessage(FacesMessage.SEVERITY_INFO, "La sesion actual se ha cerrado! ", "");
        //redirect to your login view:
        externalContext.redirect(Constantes.HOME);
}
```

4.4. Menú dinámico.

La funcionalidad para construir el menú dinámico del módulo de administración, es definida en la clase ***ControladorMenu***. La particularidad de esta funcionalidad, es que está construido programáticamente por medio de código Java en la capa de negocios del sistema y no en la capa de presentación; lo cual permite personalizarlo; según las especificaciones del requerimiento del usuario.

En el método ***agregarMenuAdministracion***, se define las instrucciones de código, para generar los submenús del módulo de administración. La lista de ítems, es construida dinámicamente en base a la información obtenida de la base de datos; de acuerdo al usuario que inicio la sesión en el sistema. Además para agrupar los datos de cada ítem necesarios para el menú y submenús, se usa la clase ***Permiso***; la cual encapsula los atributos requeridos para la construcción del menú. De esta manera, el menú es construido en base a una lista de objetos ***Permiso***, el cual tiene información del nombre del submenú así como también el nombre del archivo .xhtml (extensión del archivo jsf) al que debe ser direccionado el sistema.

En el siguiente fragmento de código se especifica el método ***agregarMenuAdministracion***, el cual se utiliza para formar el menú de acuerdo a los permisos que el usuario dispone.

Como se explico anteriormente, previo a la construcción del menú se realiza un conjunto de consultas sql que se ejecutan al momento de identificarse, la cual permite especificar cuáles son los accesos o páginas que puede observar el usuario que ha iniciado sesión y con ello, con la ayuda del método ***agregarMenuAdministracion***, se realizara la creación de cada menú ítem de acuerdo a la información consultada anteriormente.

A continuación en el método 8, se describe el que se denomina **agregarMenuAdministración**.

Método 8. agregarMenuAdministracion

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion May 08, 2014
* @fecha_ultima_actualizacionDic 12, 2014
 */

private void agregarMenuAdministracion() {
		Submenu submenu1 = new Submenu();
		submenu1.setLabel("ADMINISTRACION");
		submenu1.setIcon("ui-icon-gear");

		for (Permiso permiso : getLoginBean().getPermisos()) {
			MenuItem menuItem = new MenuItem();
			menuItem.setValue(permiso.getNombre_submodulo());
			menuItem.setUrl(permiso.getPath());
			menuItem.setIcon("ui-icon-gear");
			submenu1.getChildren().add(menuItem);
		}
		menubar.getChildren().add(submenu1);
}
```

En el método ***agregarMenuUsuario***, se definen las instrucciones de código para generar el menú para el usuario en sesión. El menú de usuario cuenta con el submenú para cerrar la sesión, este submenú permite al usuario cerrar la sesión iniciada y borrar todos los datos relacionados a la misma.

Para realizar el cierre de la sesión, el método cuenta con la instrucción de código para hacer la llamada al método logout de la clase ***LoginBean***. El llamado al método logout, se hace a través del objeto ***MethodExpression***; el cual permite realizar el llamado al código por medio de lenguaje de expresión.

En el siguiente fragmento de código se puede observar el método 9, que se denomina ***agregarMenuUsuario***, con el cual podemos crear otro menú ítem para cerrar la sesión del usuario que ha iniciado sesión como se explico anteriormente.

Método 9. agregarMenuUsuario

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion May 10, 2014
* @fecha_ultima_actualizacionDic 15, 2014
 */

private void agregarMenuUsuario() {
		Submenu userMenu = newSubmenu();
		userMenu.setLabel(getLoginBean().getPersona().getNombre() + " " + getLoginBean().getPersona().getApellido());
		userMenu.setIcon("ui-icon-person");

		MenuItem userMenuItem = new MenuItem();
		userMenuItem.setValue("Cerrar sesión");
		userMenuItem.setIcon("ui-icon-extlink");
		userMenuItem.setId("logoutItem");
		FacesContext context = FacesContext.getCurrentInstance();
		MethodExpression methodExpression = context
		.getApplication()
		.getExpressionFactory()
		.createMethodExpression(context.getELContext(), "#{loginBean.logout}", null,
		new Class[] { ActionEvent.class });
		userMenuItem.addActionListener(new MethodExpressionActionListener(methodExpression));

		userMenu.getChildren().add(userMenuItem);
		menubar.getChildren().add(userMenu);
}
```

4.5 Exportador de archivos GeoJson.

Para poder evaluar la información de la estructura de las tablas de la base de datos, se usa el set de vistas *information_schema*; que contiene la información de los objetos de la base de datos que estemos usando como: columnas, claves primarias, claves foráneas, etc.

En la funcionalidad para exportar archivos geojson, se van a evaluar dos bloques de código definidos en la capa de datos; en la clase ***ServicioGeoJson***.

El primer método ***queryTodasLasTablas***, define la sentencia sql para consultar todas las tablas que contengan campos con tipo de dato *geometry*; este tipo de dato es usado por el motor de base de datos para almacenar datos de tipo geográfico. Cabe mencionar que este tipo de dato, está disponible cuando se instala el complemento de PostGIS al motor de base de datos.

La sentencia sql, incluye una condición en donde la columna *udt_name*; que tiene la información del tipo de dato de una columna sea de tipo *geometry*. De esta manera solamente se devolverán como resultado las tablas que contengan al menos un campo de tipo geográfico, como se puede observar a continuación en el método 10:

Método 10. queryTodasLasTablas

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion May 13, 2014
* @fecha_ultima_actualizacionDic 15, 2014
 */

public static String queryTodasLasTablas() {
        return"select table_catalog, table_schema,table_name, column_name, udt_catalog, udt_schema, udt_name from
        information_schema.columns where udt_name='geometry';";
}
```

El segundo método nombrado ***queryTablasRelacionadasATablaGeografica***, define la sentencia sql para devolver las tablas relacionadas a una tabla dada, la búsqueda se hace tomando en cuenta dos premisas:

- Exista un campo que hace referencia a la clave primaria de otra tabla (clave foránea), diferente a la tabla dada.
- La clave primaria de la tabla dada está presente en otras tablas, por medio de una relación de clave foránea.

A continuación podemos observar en el método 11 la sentencia sql:

Método 11.queryTablasRelacionadasATablaGeografica

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion May 15, 2014
* @fecha_ultima_actualizacionDic 15, 2014
 */

public static String queryTablasRelacionadasATablaGeografica(String tablaGeografica)
{
        return"SELECT tc.constraint_name, tc.table_name, kcu.column_name, ccu.table_name AS foreign_table_name,
        ccu.column_name AS foreign_column_name"
        + " FROM information_schema.table_constraints AS tc JOIN information_schema.key_column_usage AS kcu ON
        tc.constraint_name = kcu.constraint_name"
        + " JOIN information_schema.constraint_column_usage AS ccu ON ccu.constraint_name = tc.constraint_name
        WHERE constraint_type = 'FOREIGN KEY' AND tc.table_name='"
        + tablaGeografica
        + "' UNION"
        + " SELECT tc.constraint_name, ccu.table_name, kcu.column_name, tc.table_name AS foreign_table_name,
        ccu.column_name AS foreign_column_name"
        + " FROM information_schema.table_constraints AS tc JOIN information_schema.key_column_usage AS kcu ON
        tc.constraint_name = kcu.constraint_name"
        + " JOIN information_schema.constraint_column_usage AS ccu ON ccu.constraint_name = tc.constraint_name
        WHERE constraint_type = 'FOREIGN KEY' AND ccu.table_name='"
        + tablaGeografica + "';";
}
```

El tercer método, define la sentencia sql para devolver la lista de campos de una tabla de la base de datos. De la misma manera, se utiliza el set de vistas *information_schema*; de la base de datos. Además, se define los campos que se necesitan sean devueltos en la consulta como son:

- ordinal_position: orden del campo en la tabla de la base de datos.
- table_name: nombre de la tala a la que pertenece el campo.
- column_name: nombre del campo.
- data_type: tipo de dato del campo.
- udt_name: tipo de dato del campo.
- is_nullable: indica si el campo acepta valores nulos o no.

A continuación se especifica el método 12, que se denomina ***queryCamposTabla*** con la sentencia especificada anteriormente.

Método 12. queryCamposTabla

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion May 20, 2014
* @fecha_ultima_actualizacionDic 16, 2014
 */

public static String queryCamposTabla(String tablaRelacionada) {
        return"SELECT ordinal_position, table_name, column_name,data_type, udt_name, is_nullable FROM
        information_schema.columns WHERE table_name='"
        + tablaRelacionada + "' ORDER BY ordinal_position;";
}
```

El método ***crearQuery***, usa el patrón de diseño Factory para generar las diferentes consultas sql. Las consultas generadas por el método, son utilizadas en el proceso de generación del archivo geojson y únicamente se genera una consulta sql; dependiendo del tipo recibido en el parámetro ***queryTipo***. Los tipos definidos para las consultas sql son:

- *QUERY_TYPE_FOR_GEOJSON*: Tipo definido para generar la consulta sql para el archivo geojson.
- *QUERY_TYPE_FOR_INSERT*: Tipo definido para generar la consulta sql realizando la inserción de datos. Esta consulta, es usada para insertar la información al ejecutar la consulta sql; en la creación de la nueva tabla en la base de datos.
- *QUERY_TYPE_FOR_GEOMETRY_TYPE*: Tipo definido para generar la consulta sql, para determinar el tipo de geometría de un campo de tipo geometry.

Dentro de la funcionalidad para generar el archivo geojson, es importante conocer la geometría del campo geográfico seleccionado por el usuario; ya que por medio de esta información se muestra las opciones de estilo propias para cada geometría. Al momento de crear una consulta sql, internamente el método hace el llamado a diferentes instrucciones de código; para generar la estructura de la consulta como es:

- Lista de campos, a ser incluidos en la consulta.
- Condiciones para filtrar la información, por medio de la clausula *Where.*

- Relaciones de tablas de las que se extraerá la información, por medio de la clausula *From.*

A continuación podemos observar el método 13 denominado crearQuery:

Método 13. crearQuery

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion May 21, 2014
* @fecha_ultima_actualizacionDic 20, 2014
 */

public static String crearQuery(List<Estructura> estructuraSeleccionadaList,
List<EstructuraPorTablaRelacionadaObject> estructuraPorTablaRelacionadaObjectList, String queryTipo,List<String>
whereList) throws SQLException {

        String query = Constantes.EMPTY;

        if (queryTipo.equals(Constantes.QUERY_TYPE_FOR_GEOJSON)) {
                query = ConstantesQuery.SELECT_STRING + " " + ConstantesQuery.SELECT_DISTINCT + " "
                + crearStringCamposParaSelectGeojson(estructuraSeleccionadaList) + " "
                + crearJoinString(estructuraPorTablaRelacionadaObjectList) + " "
                + crearWhereCondition(whereList, estructuraPorTablaRelacionadaObjectList, queryTipo);
        }
        else if (queryTipo.equals(Constantes.QUERY_TYPE_FOR_INSERT)) {

                query = ConstantesQuery.SELECT_STRING + " " + ConstantesQuery.SELECT_DISTINCT + " "
                + crearStringCamposParaInsert(estructuraSeleccionadaList) + " "
                + crearJoinString(estructuraPorTablaRelacionadaObjectList) + " "
                + crearWhereCondition(whereList, estructuraPorTablaRelacionadaObjectList, queryTipo);
        }
        else if (queryTipo.equals(Constantes.QUERY_TYPE_FOR_GEOMETRY_TYPE)) {
                query = buildQueryGeometryType(estructuraSeleccionadaList);
        }
        return query;
}
```

Parte de la funcionalidad más importante implementada, está en el método ***generateJson***; el cual permite la generación de la cadena de texto con formato geojson. Este método contiene las instrucciones de código para generar una cadena de texto geojson dinámicamente. La cadena de texto, es generada usando los atributos definidos en el proceso de la funcionalidad que son:

- **sqlForeing:** Es un atributo de tipo String, que define la consulta sql construida dinámicamente en base a las selecciones del usuario.
- **estructuraSeleccionadaList:** Es una lista de objetos de la clase Estructura, que corresponde a una lista de campos seleccionados por el usuario; de las diferentes tablas de la base de datos; para ser incluidos en el archivo geojson.
- **mapGeomProperties:** Es un mapa de objetos, que define un conjunto de datos seleccionados por el usuario correspondientes a las propiedades de estilo seleccionadas. Las propiedades de estilo (grosor, opacidad, color de relleno, etc.), son seleccionadas dependiendo del tipo de geometría (Point, Line, Polygon); seleccionada para el archivo geojson y son opcionales por lo cual este atributo puede estar vacio.

A continuación mostramos el método 14 denominado generateJson con el cual podemos generar cadenas de texto con el formato GeoJSON.

Método 14.*generateJson*

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacion May 26, 2014
* @fecha_ultima_actualizacionDic 26, 2014
 */

public static String generateJson(String sqlForeing, List<Estructura> estructuraSeleccionadaList, TreeMap<String, Object>
mapGeomProperties)
throws  SQLException {
        result = Constantes.EMPTY;
        estructuraSeleccionadaSinGeometryList = new ArrayList<Estructura>();
        result = GEOJSON;
        numeroCampos = estructuraSeleccionadaList.size();
        removeGeometryField(estructuraSeleccionadaList);
        whereCondition = buildWhereNotNullCondition(estructuraSeleccionadaList,
        existeWhereConditionInQuery(sqlForeing));
        conexion = new Conexion();
        resultSet = conexion.Consulta(sqlForeing + " " + whereCondition);
        LOG.info("..Query executed for geojson :" + sqlForeing + " " + whereCondition);
        while (resultSet.next()) {
        result += "{" + COMILLA_CHAR + "type" + COMILLA_CHAR + ":" + COMILLA_CHAR + "Feature" + COMILLA_CHAR
        + COMA_CHAR + COMILLA_CHAR + "properties" + COMILLA_CHAR + ":";
        result += buildProperties(resultSet, estructuraSeleccionadaSinGeometryList, mapGeomProperties);
        result += COMA_CHAR + COMILLA_CHAR + "geometry" + COMILLA_CHAR + ":"
        + setValorAPropiedad(resultSet, numeroCampos) + "}";
```

```
                    if (!resultSet.isLast()) {
                    result += COMA_CHAR;
                    }
            }
            result += "]}";
            resultSet.close();
            return result;
}
```

4.6 Creación de tabla en base de datos con el resultado de un query en el submódulo generador de archivos GeoJSON.

En el método ***generarTabla***, se definen las instrucciones de código; para la creación de una nueva tabla en la base de datos. Este método, hace el llamado a la capa de datos; para ejecutar un conjunto de instrucciones sql relacionadas a la creación de la nueva tabla. Como parte de los parámetros, que recibe el método para ejecutar el proceso esta:

- ***estructuraSeleccionadaList***: Lista con los campos que el usuario selecciono de las diferentes tablas.
- ***estructuraPorTablaRelacionadaObjectList***: Lista con la información de las tablas de las que se selecciono los campos para la consulta sql.
- ***whereList***: Lista de cadenas de texto, en las que se especifican las condiciones where a incluirse en la consulta sql.

Así mismo, toda la implementación de código está controlada por la clausula try/catch; de forma que si se presenta algún error al ejecutar todo el proceso el usuario será notificado en pantalla por medio de un mensaje.

A continuación se puede observar el método 15 denominado generarTabla que se utiliza para crear tablas en la base de datos con el resultado de la consulta generada en la anterior funcionalidad.

Método 15. generarTabla

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacionJun02, 2014
* @fecha_ultima_actualizacion Ene 02, 2015
 */

public void generarTabla(List<Estructura> estructuraSeleccionadaList,List<EstructuraPorTablaRelacionadaObject>
estructuraPorTablaRelacionadaObjectList, List<String> whereList) {
        try {
        ServicioNuevaTabla.ejecutarQueryTransaccionalNuevaTabla
        (
        crearQueryTabla(estructuraSeleccionadaList),
        agregarCommentNuevaTabla(),
        crearIndiceIdNuevaTabla(),

        crearInsertQuery(estructuraSeleccionadaList,
        estructuraPorTablaRelacionadaObjectList,
        whereList)
        );

        FacesContext.getCurrentInstance().addMessage(null,
        new FacesMessage(FacesMessage.SEVERITY_INFO, Mensaje.TRANSACCION_EXITOSA,""));

        clean();
        } catch (SQLException e) {
        FacesMessageUtil.addErrorMessage(Mensaje.TRANSACCION_FALLIDA + " - " + e);
        }
}
```

En la capa de datos, el método generarTabla hace el llamado al método ***ejecutarQueryTransaccionalNuevaTabla***; el cual esta implementado para ejecutar un conjunto de instrucciones en un contexto transaccional. Al ser un método que ejecuta operaciones transaccionales, se controla que ejecuten todas las operaciones definidas o ninguna; de esta manera se controla que no quede información inconsistente. Las principales operaciones sql que ejecuta este método son:

- Creación de una tabla en la base de datos.
- Adición de comentario con el detalle del usuario que creó la tabla.
- Adición de índice por la clave primaria.
- Inserción de datos en base a una consulta sql.

Dentro de la implementación del método, se definen las sentencias commit y rollback; para aplicar todos los cambios o rollback para deshacerlos en caso de fallo respectivamente.

A continuación se muestra el método 16 denominado ***ejecutarQueryTransaccionalNuevaTabla*** necesario para la creación de la nueva tabla.

Método 16. *ejecutarQueryTransaccionalNuevaTabla*

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacionJun05, 2014
* @fecha_ultima_actualizacion Ene 02, 2015
 */

public static void ejecutarQueryTransaccionalNuevaTabla(String sqlCreateTable, String sqlComment, String sqlIndex,String
sqlCreateInsert) throws SQLException {

        PreparedStatement preparedStatementCreateTable = null;

        PreparedStatement preparedStatementAddComment = null;

        PreparedStatement preparedStatementCreateIndex = null;

        PreparedStatement preparedStatementInsert = null;

        Conexion conn = new Conexion();

        Connection connection = conn.crearConexion();

        connection.setAutoCommit(false);

        preparedStatementCreateTable = connection.prepareStatement(sqlCreateTable);

        preparedStatementAddComment = connection.prepareStatement(sqlComment);

        preparedStatementCreateIndex = connection.prepareStatement(sqlIndex);

        preparedStatementInsert = connection.prepareStatement(sqlCreateInsert);

        preparedStatementCreateTable.executeUpdate();

        preparedStatementAddComment.executeUpdate();

        preparedStatementCreateIndex.executeUpdate();

        preparedStatementInsert.executeUpdate();

        connection.commit();


        if (connection != null) {

        LOG.error("La transacción se deshace.");

        connection.rollback();

        }

        closePrepareStatements(preparedStatementCreateTable, preparedStatementAddComment,
        preparedStatementCreateIndex,
```

```
            preparedStatementInsert);

            conn = null;

}
```

4.7. Descarga de archivos Geojson.

Las instrucciones de código para descargar el archivo GeoJSON, se definen en el utilitario ***DownloadUtil***. El método ***download***, del utilitario esta implementado como una operación genérica que soporta la descarga de archivos en diferente formato al especificar:

- bytesFile: arreglo de bytes de un archivo.
- contentType: tipo de contenido que va a contener el archivo a descargarse.
- fileName: nombre con el cual se va a descargar el archivo.
- extensionFile: extensión que va a tener el archivo al descargarse.

Para la funcionalidad de descarga de un archivo, se utilizan los métodos de las ***clasesFacesContext*** y ***ExternalContext***; los cuales toman la información recibida en el método. Finalmente, por medio de las operaciones de la clase abstracta ***OutputStream***; se escribe la información del archivo para que se descargue a través del navegador.

A continuación se muestra el método 17 denominado ***download*** mencionado anteriormente.

Método 17.*download*

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacionJun10, 2014
* @fecha_ultima_actualizacion Ene 05, 2015
 */

public static void download(byte[] bytesFile, String contentType, String fileName, String extensionFile) {

            try {

            FacesContext facesContext = FacesContext.getCurrentInstance();

            ExternalContext externalContext = facesContext.getExternalContext();

            externalContext.setResponseHeader("Content-Length", String.valueOf(bytesFile.length));

            externalContext.setResponseHeader("Content-Type", contentType);

            externalContext.setResponseCharacterEncoding("UTF-8");

            externalContext.addResponseCookie(Constantes.DOWNLOAD_COOKIE, "true",
```

```
            Collections.<String, Object> emptyMap());
            externalContext.setResponseHeader("Content-Disposition", "attachment;filename=" + fileName + "."
            + extensionFile);

            OutputStream outputStream = externalContext.getResponseOutputStream();

            outputStream.write(bytesFile);
            outputStream.flush();
            outputStream.close();
            facesContext.responseComplete();

            } catch (IOException e) {
                        thrownew RuntimeException("****SOME ERROR DETECTED in download:" + e);
            }
}
```

4.8 Gestión del área de influencia de una obra salesiana.

Para esta funcionalidad, se utilizó el javascript Map Toolbar de la implementación Google Maps api v3 (Google, 2013), para realizar el trazó en el mapa y además para obtener las coordenadas que posteriormente se utilizarán para generar el polígono que describe el área de influencia.

El javascript tiene en su interior algunas funciones, a continuación se observa el método 18:

Método 18.addPoint

```
/**
 * @autor Google
* @fecha_utilizacionJun 05, 2014
 */

addPoint :function(e, poly, index) {
            var e = (typeofe.latLng != "undefined")? e.latLng :e,
                        image = new google.maps.MarkerImage('images/marker-edition.png',
                        new google.maps.Size(7, 7),
                        new google.maps.Point(0, 0),
                        new google.maps.Point(5, 5)),
            imageover = new google.maps.MarkerImage('images/marker-edition-over.png',
                        new google.maps.Size(7,7),
                        new google.maps.Point(0, 0),
                        new google.maps.Point(5, 5)),
                                    path = poly.getPath(),
                                    index = (typeof index != "undefined")? index : path.length,
                                    markers = (poly.markers)? poly.markers :new google.maps.MVCArray,
            marker = new google.maps.Marker({
                                                position: e,
```

```
                                        map: map,
                                        draggable: true,
                                        icon: image


    });
varencodeString = google.maps.MVCArray;


            console.debug(e.lng());
console.debug(e.lat());
cadena=[e];

//alert(cadena);
document.getElementById('frm:j_idt16:txtTabla').value = e.lat()+'' ;
document.getElementById('frm:j_idt16:txtTabla2').value = e.lng()+'' ;
marker.index = index;
path.insertAt(index, e);
markers.insertAt(index, marker);
if(arguments[2]){
            MapToolbar.reindex(markers);
}
```

Como se puede observar en el texto anterior, el método ***addPoint*** es utilizado para añadir los puntos dentro del mapa, estos son georeferenciados a partir de la variable **e** que tiene una instancia de ***google.maps*** con el cual se puede obtener la latitud y longitud que serán utilizados en esta funcionalidad para que se genere un polígono.

Método 19.removeFeature

```
/**
 * @autor Google
 * @fecha_utilizacion Jun 05, 2014
 */

removeFeature :function(id){
        var type  = id.split('_')[0];
        var feature = MapToolbar.features[type+'Tab'][id];
          feature.$el.row.parentNode.removeChild(feature.$el.row);
        delete  MapToolbar.features[type+'Tab'][id];
        document.getElementById('frm:j_idt16:txtTabla').value = '' ;
        document.getElementById('frm:j_idt16:txtTabla2').value = '' ;
        document.getElementById('frm:j_idt16:txarea_longitud').value='';
        switch(type){
                case"placemark":
                        feature.setMap(null);
                break;
                default:
                        feature.markers.forEach(function(marker, index){
                                marker.setMap(null);
                          });
                        feature.setMap(null);
                break;
          }
        MapToolbar.select('hand_b');
},
```

El método19 denominado ***removeFeature*** se utiliza para eliminar el polígono graficado dentro del mapa, como se puede observar esta función mediante un id que se genera al momento de trazar el polígono se lo remueve todas las marcas y sus hijos del mapa, los cuales se encuentran agrupados.

Después de ser explicado las funcionalidades más relevantes del javascript utilizado, se explica a continuación las funciones más importantes, desarrolladas en el lenguaje de programación java, utilizado dentro del controlador y de los servicios para la creación de esta funcionalidad.
A continuación se muestra el método 20 denominado ***guardar***.

Método 20.Guardar

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacionJun20, 2014
* @fecha_ultima_actualizacion Ene 06, 2015
 */

public void guardar() {
        try {

        String string2 = getCadenaOculta();

        String[] parts = string2.split(",");

        String part1 = parts[0]; // 004

        if(parts.length>3){

                cadenaOculta = cadenaOculta + part1;

                cadenaOculta = "ST_GeomFromText( 'POLYGON((" + cadenaOculta + "))', 4326)";

                ServicioBeneficiario.actualizarArea(beneficiarioSelect, lugarselec, cadenaOculta);

                FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_INFO, "Area guardada", null);

                FacesContext.getCurrentInstance().addMessage(null, msg);

                limpiar();

        }else{

                FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_INFO, "Se debe escoger almenos 2
                puntos para dibujar el área.", null);

                FacesContext.getCurrentInstance().addMessage(null, msg);

        }

        } catch (Exception e) {

                FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_INFO, "Error: "+e, null);}}
```

Como se puede observar, este método que se encuentra registrado dentro del controlador, se utiliza para guardar el área de influencia graficada en el mapa.

Este método utiliza las coordenadas adicionadas mediante el javascript, a las cuales les da el formato que se utiliza para guardar en la base de datos, para que a su vez, gracias al complemento con el que cuenta la base de datos, postgis, se transformará esta sentencia en tipo geometry, para ser adicionado dentro de la tabla beneficiario.

Adicionalmente en el fragmento de código anterior se puede observar que el método que guarda el área seleccionada utiliza el método 21 denominado ***actualizarArea***, este método se describe a continuación:

Método 21. actualizarArea

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacionJun25, 2014
* @fecha_ultima_actualizacion Ene 07, 2015
 */

public static void actualizarArea(Integer idBeneficiario, Integer idLugar, String areaInfluencia) throws Exception {
        cargarDatos();
        Beneficiario modificado = buscarBen(idBeneficiario);
        if (modificado != null) {
                String sentencia = "update tb_beneficiario set areainfluencia_ben = "+ areaInfluencia+
                                                        "where id_ben ="+idBeneficiario+" and id_lug =
                "+idLugar+";";

                Conexion con = new Conexion();
                String mensaje = con.Ejecutar(sentencia);
                System.out.println(mensaje);

        } else {
                System.out.println("No existe el beneficiario que desea modificar");
                throw new Exception("No existe el beneficiario que desea modificar");
        }
}
```

Esté método básicamente se comunica con la base de datos para actualizar la tabla de beneficiario insertando dentro del campo *areainfluencia_ben*, la sentencia que se creó en el controlador y de la cual se menciono en los anteriores párrafos.

Con lo descrito en los anteriores apartados, se describe cual es el procedimiento para crear la funcionalidad de **Gestionar el área de influencia de una obra salesiana.**

4.9 Reportes.

Para el desarrollo de esta funcionalidad se utilizó el método ***generarFile***, el cual contiene las instrucciones de código para generar un archivo en base a los parámetros que recibe. Este método, usa la información recibida para que sea manipulada por las

implementaciones del API de JasperReports (Jaspersoft comunity, 2015), Como parte de las funcionalidades del API, se incluyen la lectura del archivo jasper con el diseño del reporte, el envío de datos y la generación de archivos. Una instrucción importante de este método, es el uso del objeto ***JRBeanCollectionDataSource***, por medio del cual se envía una lista de objetos que los archivos creados en JasperReport; el que puede usar como fuente de datos para llenar la información del reporte.

Al enviar la información desde este método, se garantiza que se creen nuevas conexiones a la base de datos externas a la aplicación para acceder a la información. Finalmente, el archivo generado por este método puede ser usado dentro de la aplicación del geoportal para ser descargado.

A continuación se muestra el método 22 denominado ***generarFile***, que se menciono anteriormente.

Método 22.generaFile

```
/**
 * @autorByron Sandoval - Oswaldo Tutillo
 * @departmento Desarrollo (Tesis) - Ecuador
 * @propietario Universidad Politécnica Salesiana.
 * @fecha_creacionJun22, 2014
* @fecha_ultima_actualizacion Ene 06, 2015
 */

private File generarFile(String reporteTemplate, List<T>objectDataSourceList, String fileDestino, Map<String,
Object>parameters) throws JRException {

        JasperReport reporte = (JasperReport) JRLoader.loadObjectFromFile(reporteTemplate);

        JasperPrintjasperPrint = JasperFillManager.fillReport(reporte, parameters, new
        JRBeanCollectionDataSource(objectDataSourceList));

        JRExporterexporter = new JRPdfExporter();

        exporter.setParameter(JRExporterParameter.JASPER_PRINT, jasperPrint);


        File fileTemp = new File(fileDestino);

        exporter.setParameter(JRExporterParameter.OUTPUT_FILE, fileTemp);

        exporter.exportReport();

        return fileTemp;

}
```

4.10 Cambios para utilización de base de datos de producción.

En vista que el presente proyecto es un módulo de todo el Geoportal de la Comunidad Salesiana, fue necesario realizar algunas adecuaciones a la base de datos de producción, para que podamos utilizar esta base de datos.

El objeto de utilizar la base de datos de producción es garantizar la integración con los demás módulos que conforman el sistema, para que así se garantice la integración con el mismo, además de utilizar los datos reales que utiliza el Geoportal Salesiano.

Cabe mencionar que actualmente la versión 2.0 del Módulo de Administración del Geoportal de la Comunidad Salesiana, se encuentra mayoritariamente integrado con la versión integrada del Geoportal, realizada el trabajo de grado denominado Análisis, diseño y construcción del módulo de seguridad e integración de los módulos de visualización y gestión de datos geográficos del geoportal de la comunidad salesiana versión 2.0, realizado por los señores Torres Carina & Aldaz Darwin, los cuales integraron la versión 2.0 del módulo de Visualización realizada por los señores Cofre Víctor y Toledo Stalin, salvo dos funcionalidades las cuales se detalla a continuación:

- Exportador de archivos GeoJson
- Reportes

A continuación se encuentran detallados los cambios más relevantes en la base de datos que fueron necesarios para que este nuevo módulo utilice la base de datos que utiliza actualmente el Geoportal de la Comunidad Salesiana:

1. Agregar tabla parámetros: El módulo de Administración versión 2.0, utiliza algunos parámetros desde la base de datos, es por ello que fue necesario crear esta tabla en la base de datos de producción de la siguiente manera:

```
CREATE TABLE tb_parametros_proyecto
(
id serial NOT NULL, -- pk de la tabla de parametros
nombre character(60), -- nombre del parametro del sistema
valor character(150), -- valor del parametro
 CONSTRAINT pk_parametros_proyecto PRIMARY KEY (id )
)
```

```
WITH (
 OIDS=FALSE
);
```

2. Poblar datos en tabla parámetros: Una vez creada la tabla en la versión de producción, es necesario poblar de datos la misma para lo cual se lo debe realizar de la siguiente manera:

```
INSERT INTO tb_parametros_proyecto(nombre, valor) VALUES ('path_imagenes_tmp','/opt/TESIS-TUTILLO-SANDOVAL/geoportal/tmp');
INSERT INTO tb_parametros_proyecto(nombre, valor) VALUES ('pathArchivosTemp','/opt/TESIS-TUTILLO-SANDOVAL/geoportal/tmp');
INSERT INTO tb_parametros_proyecto(nombre, valor) VALUES ('reportNombreInstitucion','Universidad Politécnica Salesiana');
INSERT INTO tb_parametros_proyecto(nombre, valor) VALUES ('reportNombreSistema','Geoportal de la Comunidad Salesiana');
INSERT INTO tb_parametros_proyecto(nombre, valor) VALUES ('reportPathLogo','/opt/TESIS-TUTILLO-SANDOVAL/geoportal/pdf/logoUPS.png');
INSERT INTO tb_parametros_proyecto(nombre, valor) VALUES ('reportObraSalesianaTemplate','/opt/TESIS-TUTILLO-SANDOVAL/geoportal/jasper/reportObra.jasper');
INSERT INTO tb_parametros_proyecto(nombre, valor) VALUES ('reportCasaSalesianaTemplate','/opt/TESIS-TUTILLO-SANDOVAL/geoportal/jasper/reportCasa.jasper');
INSERT INTO tb_parametros_proyecto(nombre, valor) VALUES ('reportPathBackground','/opt/TESIS-TUTILLO-SANDOVAL/geoportal/pdf/background.png');
INSERT INTO tb_parametros_proyecto(nombre, valor) VALUES ('imagenesLugaresPath','/opt/TESIS-TUTILLO-SANDOVAL/geoportal/imagenes/FotosLugaresSalesianos/');
```

Con estos cambios más el poblado de datos de las tablas tb_perfil_tb_submodulo, tb_permiso y tb_submodulo, se logró utilizar la base de datos de producción y con ello, como se especifico anteriormente, quedo demostrado que esta nueva versión se puede integrar con los demás módulos que conforman el Geoportal de la Comunidad Salesiana.

CAPÍTULO 5
IMPLEMENTACIÓN

En el presente capítulo, se describe la implementación del módulo de administración del Geoportal Salesiano en su versión 2.0.

Debido a que esta versión corresponde a un módulo de todo el Geoportal Salesiano, se realizará la implementación siguiendo las mismas especificaciones de las versiones implementadas anteriormente para este Geoportal.

5.1 Implementación.

Este módulo se implementó en un servidor HP ProLiant ML110 G7, ubicado dentro de las instalaciones de la universidad y el mismo que tiene las siguientes características:

Tabla 21. *Características de software*

Especificaciones de software	
Sistema Operativo	Centos versión 5.9
Base de Datos	PostgreSQL versión 9.1.9
Datos Espaciales	PostGIS versión 1.5
Servidor Web	Apache 2.2.24
Lenguaje de Desarrollo	Java 7

Nota. Cuadro con características de software del servidor.

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Tabla 22. *Características de hardware*

Especificaciones de hardware	
Procesador	1 x Intel Xeon E3-1220 / 3.1 GHz (Quad-Core)
Memoria	2 GB (instalados) / 16 GB (máx.) - DDR3 SDRAM - ECC - 1333 MHz - PC3-10600

Controlador gráfico	Matrox G200
Conexión de redes	Adaptador de red - Ethernet,Fast Ethernet,Gigabit Ethernet - Puertos Ethernet:2 x Gigabit Ethernet
Disco duro	Disco duro 1 x 250 GB - estándar - Serial ATA-300

Nota. Cuadro con características de hardware del servidor.

Elaborado por: Sandoval Byron &Tutillo Oswaldo

5.2 Restauración de la base de datos.

Para realizar la restauración de la base de datos en el servidor se debe realizar los siguientes pasos:

1.Ingreso de usuario postgres.

Primero se debe de ingresar a la consola de postgres en un terminal, para esto hay que ingresar en el terminal con el comando: psql –U postgres, una vez ahí nos solicitara los datos de conexión.

Una vez ingresado los datos de conexión es decir el usuario y el password de postgres podremos ingresar a la base de datos.

A continuación en la figura 26 se muestra el terminal con el ingreso del usuario Postrgres para conectarse a la base de datos.

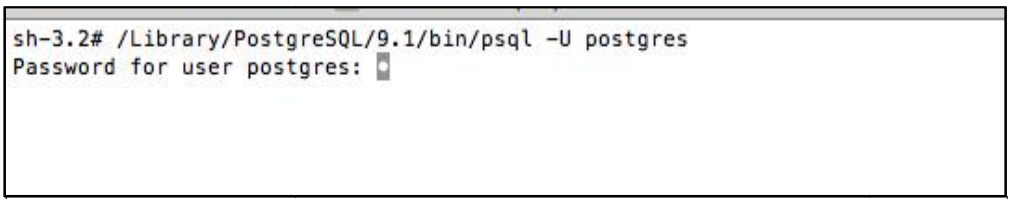


*Figura 26.*Ingreso de usuario Postgres por medio de un terminal

Elaborado por: Sandoval Byron &Tutillo Oswaldo

2. Creación de base de datos.

Para crear la base de datos, se debe ingresar la sentencia sql CREATE DATABASE seguido del nombre de la base de datos que le deseemos dar a nuestra nueva base, por medio del terminal en el que ya nos encontramos conectados a la base de datos postgres.

A continuación en la figura 27 se muestra el terminal con el ingreso del comando CREATE DATABASE para la creación de la base de datos.

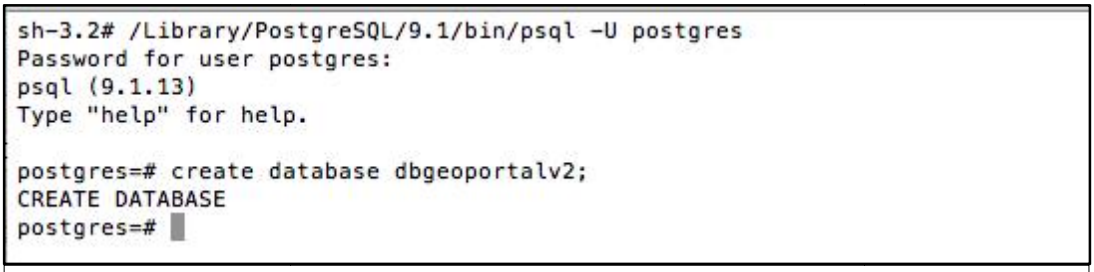


Figura 27. Creación de base de datos en Postgres por medio de un terminal

Elaborado por: Sandoval Byron &Tutillo Oswaldo

3. Incorporación de tablas para datos espaciales.

Después de realizado lo anterior debemos adecuar nuestra base de datos para que permita el ingreso de datos espaciales en ella.

Para realizar esta tarea cargamos el script de postgis.sql que se encuentra disponible en la carpeta de postgres cuando descargamos el plugin de nuestra base de datos.

Para correr el script nos dirigimos a donde se encuentra el script e ingresamos la siguiente sentencia: PSQL –U POSTGRES –D NOMBRE_BASE –F POSTGIS.SQL

Con esta sentencia cargamos el script de postgis.sql el cual crea dos tablas geometry_columns y spatial_ref_sys que servirán para albergar datos geográficos.

A continuación en la figura 28 se muestra el terminal con el ingreso del comando para cargar el script postgis.sql.

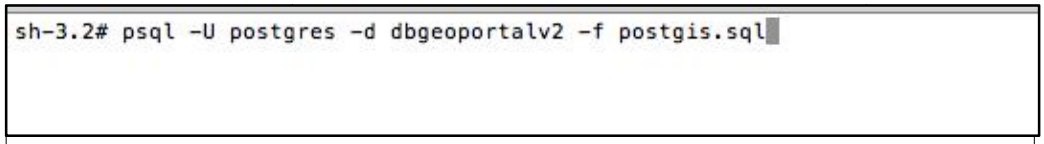


Figura 28. Sentencia para creación de tablas espaciales

Elaborado por: Sandoval Byron &Tutillo Oswaldo

4. Cargar datos en las tablas espaciales.

Después de realizado lo anterior debemos cargar la información espacial referencial en las tablas creadas anteriormente. Para ello en nuestra base de datos haremos correr el script spatial_ref_sys.sql, el cual contiene datos geográficos mundiales (puntos o coordenadas) conocimos como información EPSG. Con ello permitirá a nuestra base ingresar datos geográficos por medio de coordenadas de longitud y latitud.

Para correr el script nos dirigimos a donde se encuentra el script e ingresamos la siguiente sentencia: PSQL –U POSTGRES –D NOMBRE_BASE –F SPATIAL_REF_SYS.SQL

Con esta sentencia cargamos la información EPSG a las tablas creadas anteriormente convirtiendo así a nuestra base de datos en una espacial.

A continuación en la figura 29 se muestra el terminal con el ingreso del comando para cargar el script spatial_ref_sys.sql.

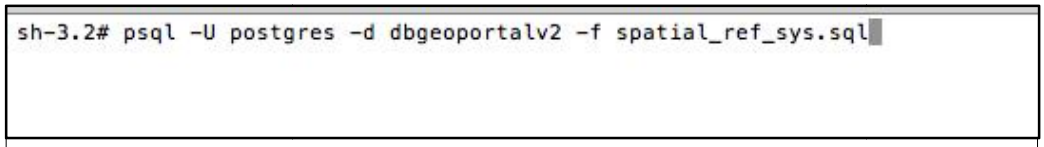


Figura 29. Sentencia para cargar información EPS

Elaborado por: Sandoval Byron &Tutillo Oswaldo

5. Restaurar base de datos.

Una vez realizado lo anterior ya tenemos lista nuestra base de datos para que ingrese datos geográficos.

Para correr el script nos dirigimos a donde se encuentra el script que deseemos cargar e ingresamos la siguiente sentencia: PSQL –U POSTGRES –D NOMBRE_BASE –F NOMBRE_DEL_SCRIPT.SQL

A continuación en la figura 30 se muestra el terminal con el ingreso del comando para cargar el script DbGeoportalSalesiano.sql.

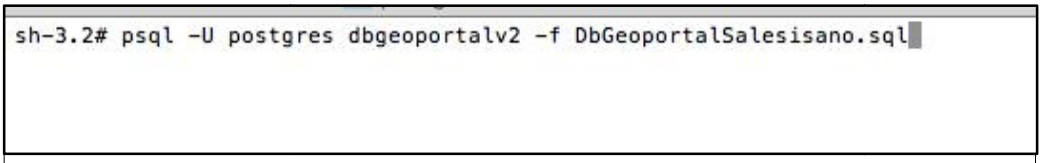


Figura 30. Sentencia para correr el script DbGeoportalSalesiano.sql

Elaborado por: Sandoval Byron &Tutillo Oswaldo

Con estos pasos hemos logrado restaurar la base de datos, que almacenara la información del Geoportal Salesiano.

5.3 Configuración y carga del archivo .war en el servidor Apache Tomcat.

Para realizar la configuración del servidor y cargar el aplicativo al mismo, se debe realizar los siguientes pasos:

1. Creación de un usuario.

En vista que por defecto Apache Tomcat no tiene configurado un usuario que le permita entrar al manager webapp del server, por lo cual se debe crear un usuario con el rol manager-gui, está configuración se hace en el archive tomcat-users.xml que está en: /$APACHE_TOMCAT_HOME/conf.

En la figura 31 podemos observar donde se encuentra escrito el usuario Tomcat con el rol manager-gui, el cual necesitamos para realizar la implementación del archivo war.

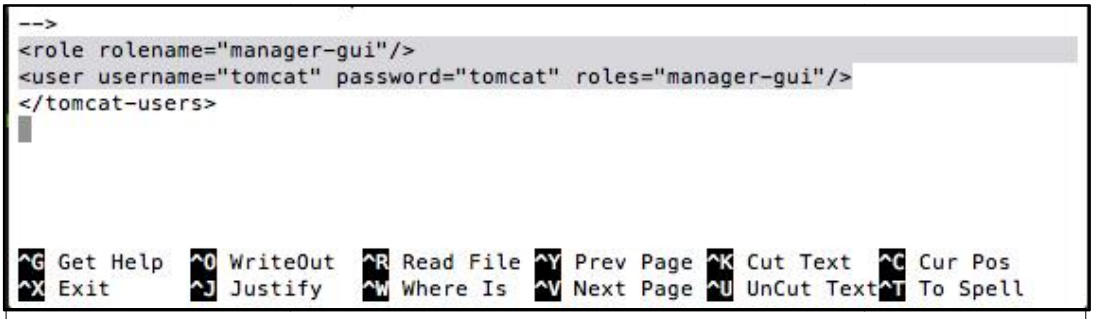


Figura 31. Creación de usuario en Apache Tomcat

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

2.Aumento de memoria del servidor.

Después de realizar la creación del usuario para el servidor se debe aumentar la memoria para que pueda desplegar el aplicativo, para ello se debe aumentar la capacidad máxima del servidor o (MAX FILE SIZE) en inglés, que por defecto en Apache Tomcat 7 viene configurado para cargar archivos de 50 MB.

Para aumentar la capacidad se debe configurar el archivo: /$APACHE_TOMCAT_HOME/webapps/manager/WEB-INF

Que para nuestro caso se aumentó a 200 MB para cargar el aplicativo.

Esto se puede observar en la figura 32 que se muestra a continuación:

Figura 32. Aumento de memoria del servidor

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

3. Iniciamos servidor Apache.

Ya realizada la configuración se debe iniciar el servidor, para ello nos dirigimos a

/$APACHE_TOMCAT_HOME/bin y ejecutamos el comando ./startup.sh, con ello se levantara el servidor con las configuraciones iniciales.

Esto se puede observar en la figura 33 siguiente:

```
sh-3.2# cd bin/
sh-3.2# ./startup.sh
Using CATALINA_BASE:   /Users/Ozwald/Documents/java/servers/apache-tomcat-7.0.57
Using CATALINA_HOME:   /Users/Ozwald/Documents/java/servers/apache-tomcat-7.0.57
Using CATALINA_TMPDIR: /Users/Ozwald/Documents/java/servers/apache-tomcat-7.0.57
/temp
Using JRE_HOME:        /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Content
s/Home
Using CLASSPATH:       /Users/Ozwald/Documents/java/servers/apache-tomcat-7.0.57
/bin/bootstrap.jar:/Users/Ozwald/Documents/java/servers/apache-tomcat-7.0.57/bin
/tomcat-juli.jar
Tomcat started.
sh-3.2#
```

Figura 33. Inicio de servidor Apache

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

4. Ingreso al servidor.

Con la configuración realizada anteriormente y como ya se encuentra inicializado el servidor podemos ingresar al servidor.

Para ingresar al servidor debemos ingresar a un navegador o browser y digitar la dirección http://ide.ups.edu.ec:8080. Cabe destacar que el puerto por defecto que se encuentra configurado en Apache es 8080 y por ello se levantara la página en este puerto.

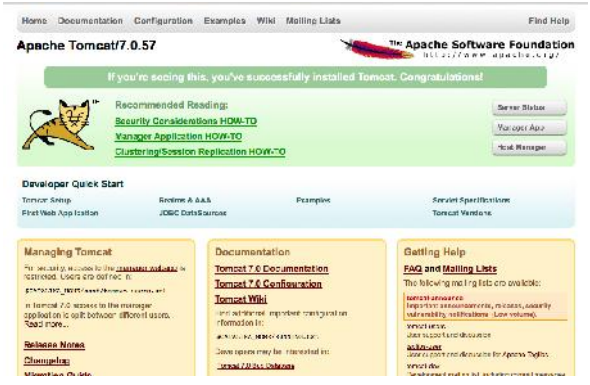


Figura 34. Pantalla principal de Apache Tomcat

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

5. Ingreso a la página Manager Webapp.

Una vez ingresado en el servidor, debemos seleccionar la opción Manager Webapp, la cual nos permite administrar todas las aplicaciones que se encuentran deployadas en el servidor. Para ingresar en esta sección nos solicita un usuario y una contraseña la cual configuramos previamente, como se indicó en los anteriores párrafos.

En la figura 35 que se muestra a continuación observamos la pantalla de ingreso para esta sección.

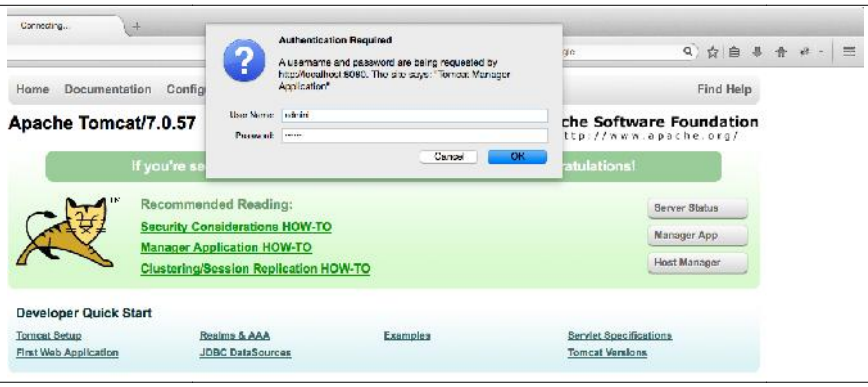


Figura 35. Acceso sección Webapp de Apache Tomcat

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

6. Ingreso a la sección Deploy.

Después de realizado lo anterior, se nos aparecerá la pantalla principal del Manager Webapp, en la cual nos dirigiremos hacia la parte inferior de la página donde se encuentra la sección de Deploy.

En la figura 36 que se muestra a continuación observamos la pantalla Manager Web App, con la sección Deploy.

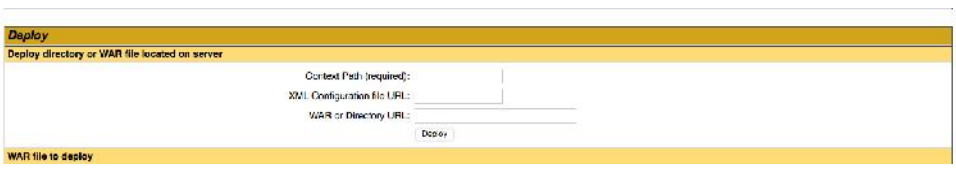


Figura 36. Página Manager Webapp con la sección Deploy

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

5. Carga de archivo .war

Continuando con nuestra explicación, en la sección Deploy de la página Manager Webapp, podemos subir aplicativos al servidor de algunas maneras, para nuestro caso lo realizaremos seleccionando el archivo .war que genera el ID eclipse.

Para realizar la subida o deploy de este archivo, se tiene que pulsar el botón *browse* que se encuentra en esta sección de la página y seleccionar el archivo .war que deseemos subir al servidor, para nuestro caso será el archivo prjAdministracion.war y después presionar el botón deploy.

En la figura 37 que se muestra a continuación observamos la sección deploy de la página Manager Webappp, una vez que se ha procedido a subir el archivo prjAdministracion.war.

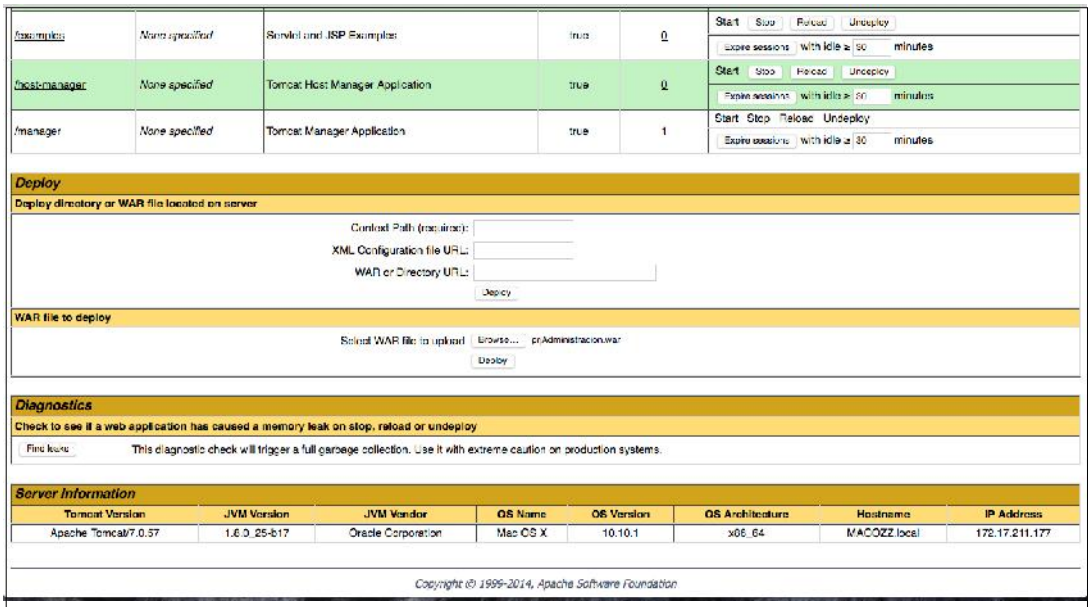


Figura 37. Sección Deploy cargado el archivo prjAdministracion.war.

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

6. Desplegar (deployar) el aplicativo en el servidor.

Una vez realizado lo anterior, se tiene que esperar que se suba todo el archivo.war al servidor, el tiempo de espera para completar esta acción dependerá del peso del archivo.

Después de haber concluida esta tarea se nos indica que el archivo ha sido subido o deployado sin ningún inconveniente y se enlistara el nombre de nuestro proyecto en la tabla principal de la página Manager webapp; con ello ya se encontrar nuestro aplicativo deployado en el servidor listo para ser utilizado en cualquier momento con la ayuda de un navegador.

En la figura 38 que se muestra a continuación se observa el listado de la página Manager Webapp incluido el aplicativo prjAdministracionV2.

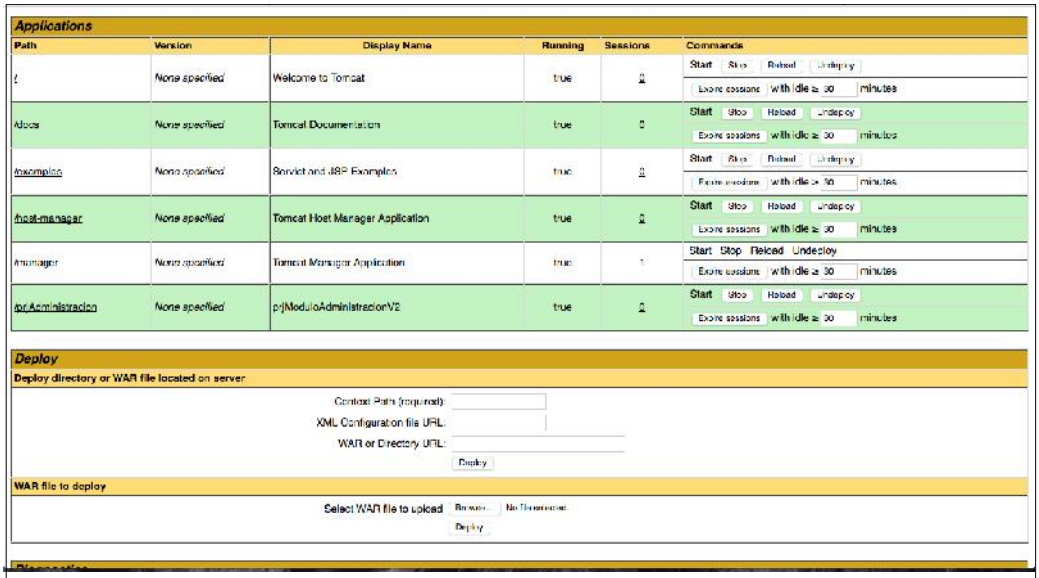


Figura 38. Página Manager Webapp con el aplicativo prjAdministracionV2.

Elaborado por: Sandoval Byron &Tutillo Oswaldo.

Con los pasos anteriores hemos realizado la implementación de nuestro módulo en el servidor apache provisto por CIMA.

CONCLUSIONES

- Para cumplir con la creación de los nuevos requerimientos para esta nueva versión del módulo de administración, fue necesario modificar la estructura de datos que se entregó en un principio, con ello se pudo desarrollar las funcionalidades de seguridad, control de acceso a los usuarios y además potenciar la base de datos y en si todo el Geoportal Salesiano.

- Para que la aplicación tenga una estructura ordenada, se definieron las capas en base al modelo MVC (Modelo, Vista y Controladores). Esta práctica nos ayudó a organizar de una mejor manera el código y además conseguimos que nuestro proyecto pueda ser escalable y que los cambios o nuevos requerimientos sigan un estándar.

- Para recolectar la información geográfica, fue necesaria la utilización de dispositivos que permiten la captura de coordenadas espaciales (GPS). Con estos dispositivos se pudo capturar las trazas de las obras salesianas.

- Para recolectar de mejor manera los requerimientos de la nueva versión del módulo de administración se elaboraron historias de usuario y maquetas de diseño de las nuevas interfaces. Esta práctica, facilitó la comprensión de los requerimientos y el desarrollo en el código así como el diseño final de las funcionalidades.

- En el proceso de elaboración del producto final, se desarrollaron funcionalidades pequeñas y utilizables para el cliente. Esto permitió que se tenga una continua y oportuna retroalimentación, en varios aspectos como diseño, funcionamiento, entre otros. Permitiendo la construcción de una aplicación más apegada a las necesidades del cliente. También como ventaja de esta práctica, se puede rescatar que la elaboración de funcionalidades pequeñas ayudó a que la integración sea continua y mucho más fácil.

- Por medio del lenguaje de modelado UML, se establecieron los modelos de clases Java usados en la aplicación, con ello logramos visualizar de mejor manera cuales son los requisitos reales del cliente.

- Para verificar cada uno de los métodos, se desarrollaron un set de pruebas unitarias, para verificar el funcionamiento del código de la aplicación. Con esta práctica comprobamos que cada uno de los nuevos métodos relevantes funcionan y cumplen el objetivo con el que fueron creados.

- Para la realización del presente trabajo de tesis, se requirió el trabajo conjunto con estudiantes de otras tesis, asociadas al Geoportal de la Comunidad Salesiana. Bajo esta modalidad de trabajo, se requirió que la información sea compartida continuamente (base de datos, código Java, documentos, etc.), en el servicio de almacenamiento en la nube Dropbox, esta experiencia permitió que el Geoportal se construya de manera paralela, logrando así no retrasarnos en el desarrollo de la nueva versión de todo el Geoportal y compartiendo con los demás estudiantes las experiencias, enriqueciendo también nuestro módulo.

RECOMENDACIONES

- Dada la continua necesidad de compartir información, se recomienda el uso de una herramienta de control de versiones para el código. Una opción apropiada es Subversion (SVN), un software open source, para el versionamiento de código que puede ser administrada fácilmente y ofrecer un control más confiable de las versiones de los proyectos de tesis.

- Dado el uso de herramientas Open Source, para el desarrollo de la aplicación; se facilita el acceso a las versiones más recientes de las herramientas; por lo cual se sugiere realizar la actualización de:
  - Componentes de Primefaces.
  - Motor de base de datos PostgreSQL.
  - Módulo de manejo de datos espaciales Postgis.
  - Versión de Java de la aplicación.

- Debido al crecimiento del Geoportal Salesiano, el mismo necesita una plataforma de alto rendimiento con características de encolamiento, transaccionalidad, control de acceso para objetos distribuidos y control de concurrencia, además que sea de código abierto; estas características se las puede encontrar en un servidor JBoss, es por ello que se recomiendo su uso para las posteriores versiones del Geoportal.

- Se recomienda la integración final de la versión 2.0 del Módulo de Administración con las demás versiones, para que así se pueda tener un sistema consolidado.

LISTA DE REFERENCIAS

Barcelona, A. d. (s.f.). *Geoportal bcn*. Recuperado el 19 de junio de 2013, de http://www.bcn.cat/geoportal/es/geoportal.html

Calle, F. (2013). *slideshare.* Recuperado el 20 de junio de 2013, de http://www.slideshare.net/Decimo/arquitectura-3-capas

Castellano, P. e. (2013). *La tecnología JavaServer Faces*. Recuperado el 19 de junio de 2013, de http://www.programacion.com/articulo/introduccion_a_la_tecnologia_javaserver_faces_233

DEFINICIONDE. (2013). *definicion.de*. Recuperado el 20 de junio de 2013, de http://definicion.de/seguridad-informatica/

*Diagramas de Casos de Uso*2011

ehowenespano. (2013). *ehowenespano*. Recuperado el 20 de junio de 2013, de http://www.ehowenespanol.com/son-bases-datos-espaciales-info_254596/

EURORESIDENTES. (2013). *EURORESIDENTES*. Recuperado el 20 de junio de 2013, de http://www.euroresidentes.com/gps/que-es-el-gps.htm

Fernández, M. R. (2013). *Proceso de Georeferenciación de la Cartografía Histórica*. Recuperado el 19 de junio de 2013, de http://www.expobus.us.es/cartografia/salas/sala12/georreferenciacion.html

Google. (31 de enero de 2013). *Google developers*. Recuperado el 15 de Agosto de 2014, de https://developers.google.com/maps/documentation/javascript/tutorial?hl=es#Loading_the_Maps_API

Gracia, L. M. (2013). *Un poco de Java*. Recuperado el 19 de junio de 2013, de http://unpocodejava.wordpress.com/2013/03/06/que-es-geojson/

Jaspersoft comunity. (2015). Recuperado el 12 de 2014, de JasperReports® Library: https://community.jaspersoft.com/project/jasperreports-library

MASADELANTE. (2013). *MASADELANTE.COM*. Recuperado el 19 de junio de 2013, de http://www.masadelante.com/faqs/base-de-datos

OFBiz, B. (2014). *Capítulo 5. Metodología*. Recuperado el 2014, de http://oness.sourceforge.net/proyecto/html/ch05.html

procesos de software. (2013). Recuperado el 20 de junio de 2013, de http://procesosdesoftware.wikispaces.com/METODOLOGIA+XP

Stalin, C. V. (2014). *Tesis de grado.* Quito.

GLOSARIO DE TÉRMINOS

ADV: (Vista de Datos Abstracta) se utiliza para representar la información de una forma gráfica.

Deployar: Palabra en inglés que significa desplegar.

Framework: Marco de trabajo, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular.

GeoJSON: Formato de archivo ligero de intercambio de datos que se utiliza para información geográfica.

HTML: Hyper Text Markup Language (lenguaje de marcas de hipertexto), sirve para la elaboración de páginas web.

JSF: Framework para aplicaciones java.

JSON: (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

URL: Siglas en inglés de uniform resource locator (en español localizador uniforme de recursos).

ANEXOS

Anexo1: Historia de usuario Menú dinámico

Historia de Usuario No.	A1.1
Nombre de la historia de usuario:	Menú de administración dinámico
Elaborado por:	Sandoval Byron &Tutillo Oswaldo
Fecha de elaboración:	23de octubre de 2014
Usuario:	Administrador general, Administrador GIS, Administrador datos
Como Administrador general, Administrador GIS, Administrador datos, necesito disponer de un menú de usuario de acuerdo a los permisos otorgados; para poder navegar entre las diferentes funcionalidades del modulo de administración así como las funcionalidades disponibles del usuario que inicio sesión en el sistema.	
Criterios de aceptación: El menú debe estar ubicado horizontalmente debajo de la imagen de la cabecera de la página del sistema. El menú siempre debe estar visible para el usuario. Los submenús deben de estar agrupados y disponibles en cada menú si aplica. **El sistema debe permitir al usuario realizar lo siguiente:** Cuando el usuario ingresa al sistema se muestra la pantalla de inicio con el menú y las opciones: Home: Re-direcciona la pagina actual hacia la página inicial del sistema. Login: Habilita la ventana emergente para el inicio de sesión. Si el usuario inicia la sesión en el sistema, el menú muestra las opciones: Nombre de usuario que inicio sesión: En esta opción de menú se debe mostrar el nombre y el apellido del usuario que inicio la sesión en el sistema con el submenú Cerrar Sesión. Home: Esta opción de menú permite al usuario retornar inmediatamente a la página principal del sistema.	

Anexo2: Historia de usuario iniciar sesión en el sistema - Login.

Historia de Usuario No.	A1.2
Nombre de la historia de usuario:	Iniciar sesión en el sistema - Login
Elaborado por:	Sandoval Byron &Tutillo Oswaldo
Fecha de elaboración:	23 de octubre de 2014
Usuario:	Administrador general, Administrador GIS, Administrador datos

Como Administrador general, Administrador GIS, Administrador datos, necesito disponer de una funcionalidad para iniciar la sesión en el sistema y acceder a las funcionalidades disponibles.

Criterios de aceptación

- El sistema no debe permitir a los usuarios ingresar a alguna funcionalidad si primero no ha iniciado la sesión.

Prerrequisitos

La siguiente información, debe de estar ingresada en la base de datos utilizada por el sistema.

- Nombres de usuario activos en el sistema
- Registros de menús y submenús del sistema.
- Registros de permisos de usuario.

El sistema debe permitir al usuario realizar lo siguiente

- Cuando el usuario hace clic el menú Login, el sistema:
  - Muestra la pantalla emergente para el ingreso del nombre de usuario, contraseña y el botón Login para verificar los datos.
- Si el usuario hace clic en el botón Login de la ventana emergente y la información ingresada es correcta, el sistema:
  - Recarga la página principal y el menú con las siguientes opciones:
    - ✓ Nombre de usuario que inicio sesión: En esta opción de menú, se debe mostrar el nombre y el apellido del usuario que inicio la sesión en el sistema.
    - ✓ Home: Esta opción de menú permite al usuario retornar inmediatamente a la página principal del sistema.
    - ✓ Administración: Este menú, contiene todos los submenús de acuerdo a

los permisos que el usuario tenga en la base de datos. • Cuando el usuario hace clic en cualquier submenú; el sistema re-direcciona la página actual a la pantalla de la funcionalidad seleccionada.

Anexo3: Historia de Usuario Cerrar sesión – Logout.

Historia de Usuario No.	A1.3
Nombre de la historia de usuario:	Cerrar Sesión - Logout
Elaborado por:	Sandoval Byron &Tutillo Oswaldo
Fecha de elaboración:	23 de octubre de 2014
Usuario:	Administrador general, Administrador GIS, Administrador datos
Como Administrador general, Administrador GIS, Administrador datos, necesito disponer de una funcionalidad para cerrar la sesión iniciada en el sistema cuando ya no se requiera continuar usando el sistema.	
Criterios de aceptación: • Esta funcionalidad está disponible para todos los usuarios que inician sesión en el sistema. • Esta funcionalidad, limpia toda la información del usuario y aquella generada en cualquiera de las funcionalidades del modulo de administración. **Prerrequisitos** • Debe de existir una sesión de usuario activa. **El sistema debe permitir al usuario realizar lo siguiente:** • Cuando el usuario, hace clic en el submenú Cerrar Sesión del menú de usuario, el sistema: o Limpia los datos de sesión del usuario y los generados en las funcionalidades a las cuales haya accedido. o Re-direcciona el sistema a la página de inicio del sistema.	

Anexo 4: Historia de usuario Georeferenciación área de influencia.

Historia de Usuario No.	A2.1
Nombre de la historia de usuario:	Georeferenciación Área de influencia

Elaborado por:	Sandoval Byron & Tutillo Oswaldo
Fecha de elaboración:	23 de octubre de 2014
Usuario:	Administrador GIS

Como Administrador GIS, necesito disponer de una utilidad parageoreferenciar un polígono en un mapa interactivo; para determinar el área de influencia de una obra salesiana en el país.

Criterios de aceptación:

- Esta funcionalidad debe de contar con un mapa interactivo en el cual se pueda graficar fácilmente un polígono.
- Se debe de permitir borrar un polígono y volver a crear uno nuevo según la necesidad del usuario.

Prerrequisitos

La siguiente información, debe de estar ingresada en la base de datos utilizada por el sistema.

- Casas Salesianas.
- Obras Salesianas.
- Lugares.

El sistema debe permitir al usuario realizar lo siguiente:

- Cuando el usuario, hace clic en el submenú Generador Área de influencia , el sistema:
  - Muestra la pantalla para generar él un área de influencia, con los siguientes elementos:
  - Mapa interactivo para graficar un polígono.
  - Botón para crear un nuevo polígono en el mapa.
  - Listas desplegables para seleccionar: Casa Salesiana, Obra Salesiana y Lugar.
  - Botón guardar y Cancelar.
- Cuando el usuario hace clic en el botón para crear un nuevo polígono, el sistema:
  - Habilita la creación de puntos de georeferencia en el mapa cuando el usuario hace clic en el mapa.

    Nota: Cuando el usuario georeferencia más de un punto, la funcionalidad grafica una línea entre los puntos existentes, para que el usuario pueda

visualizar el polígono.

- Cuando el usuario selecciona la información de las listas desplegables, el sistema:
  - Carga la información del siguiente elemento en base a la selección.
- Cuando el usuario hace clic en el botón Guardar, el sistema:
  - Verifica que exista información seleccionada en las listas desplegables (Casa Salesiana, Obra Salesiana y Lugar).
  - Verifica que exista un graficado un polígono valido en el mapa interactivo. Para que el polígono ingresado por el usuario sea valido:
    - ✓ El polígono debe de tener más de dos puntos en el mapa interactivo.
  - Guarda el registro en la base de datos y muestra el mensaje de confirmación.
  - Limpia la información previa ingresada por el usuario.
- Si el usuario, hace clic en el botón Cancelar, el sistema:
  - Limpia la información previa ingresada.

Anexo*5*. Diccionario de datos implementado en versiones anteriores

Diccionario de datos tabla casa Salesiana

Nombre	tb_casasalesiana			
Descripción	Almacena la información básica de las Casas Salesianas			
Primary Key	id_cas			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_cas	serial	Yes	Identifica a la casa a la cual hace referencia
	nombre_cas	text	Yes	Nombre de la casa salesiana
	direccion_cas	text	Yes	Dirección o ubicación de la casa salesiana.
	telefono_cas	text	No	Teléfono de la casa salesiana.
	correo_cas	text	Yes	Correo de la casa salesiana
	director_cas	text	Yes	Director o persona responsable de la casa salesiana.
	pathicono_cas	text	Yes	Es el directorio o path donde se encuentran almacenados todos los iconos disponibles para las casas salesianas.
	nombrecorto_cas	text	Yes	Es una abreviación del nombre de la casa salesiana por motivo

				de diseño de interfaz.
	estado_cas	boolean	No	Almacenará un true o false

Nota. Diccionario de datos de la tabla tb_casasalesiana

Elaborado por: Cofre Víctor & Toledo Stalin

Diccionario de datos tabla tipo de obra salesiana

Nombre	tb_tipoobra			
Descripción	Almacena la información básica del tipo de obra			
Primary Key	id_tobr			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_tobr	serial	Yes	Identificación única del tipo de obra
	descripcion_tobr	text	Yes	Descripción del tipo de obra
	pathicono_tobr	text	Yes	Se especifica el directorio en el cual se guardaran los iconos disponibles para los tipos de obra.
	estado_tobr	boolean	No	Almacenará un true o false

Nota. Diccionario de datos de la tabla tb_tipoobra

Elaborado por: Cofre Víctor & Toledo Stalin

Diccionario de datos tabla lugar

Nombre	tb_lugar			
Descripción	Almacena la información del lugar con énfasis en las coordenadas para la obra salesiana asociada a este.			
Primary Key	id_lug			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_lug	serial	Yes	Identificador único del lugar
FK	id_obr	integer	Yes	Representa al indicador de la obra
FK	id_elug	integer	Yes	Representa al indicador del estilo de un lugar.
	nombre_lug	text	Yes	Es el nombre del lugar.
	descripcion_lug	text	Yes	Es la descripción del servicio de la obra
	responsable_lug	text	Yes	Responsable de la obra asociada a este lugar.
	direccion_lug	text	Yes	Dirección del lugar.
	telefono_lug	text	Yes	Número telefónico del lugar.
	coordenada_lug	geometry (puntos	Yes	Es la ubicación geográfica del

		ymultipuntos)		lugar
	estado_lug	boolean	No	Almacenará un true o false

Nota. Diccionario de datos de la tabla tb_lugar

Elaborado por: Cofre Víctor & Toledo Stalin

Diccionario de datos tabla obras salesiana

Nombre	tb_obrasalesiana			
Descripción	Almacena la información de las Obras Salesianas			
Primary Key	id_obr			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_obr	serial	Yes	Identificador único de la obra salesiana a la cual hace referencia
FK	id_tobr	integer	Yes	Representa al indicador del tipo de obra
FK	id_cas	integer	Yes	Representa al indicador de la casa
	denominacion_obr	text	Yes	Es el nombre de la obra salesiana.
	camposervicio_obr	text	Yes	Es el área de servicio de la obra salesiana
	productos_obr	text	Yes	Son los servicios que ofrece la obra salesiana a la comunidad.
	horario_obr	text	Yes	Es el horario de atención de la obra salesiana.
	informacion_obr	text	Yes	Es la descripción, historia e información de la obra salesiana.
	pathicono_obr	text	Yes	Se especifica el directorio en el cual se guardaran los iconos disponibles para las obras salesianas.
	nombrecorto_obr	text	Yes	Es una abreviación del nombre de la obra salesiana
	paginaweb_obr	text	No	Es la dirección web de la obra salesiana
	estado_obr	boolean	No	Almacenará un true o false

Nota. Diccionario de datos de la tabla tb_obrasalesiana

Elaborado por: Cofre Víctor & Toledo Stalin

Diccionario de datos tabla estilo beneficiario

Nombre	tb_estilobeneficiario			
Descripción	Almacena la información de los diferentes estilos que tendrá el beneficiario			
Primary Key	id_eben			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_eben	serial	Yes	Identificador único del estilo que tendrá un beneficiario.
	descripcion_eben	text	Yes	Es una pequeña frase que indica cuales son los beneficiarios
	colorBorde_eben	text	Yes	Es el color que tendrá el borde del área de influencia
	opacidadBorde_eben	text	Yes	Es la intensidad de color que tendrá el borde del área de influencia
	grosorBorde_eben	text	Yes	Es el grosor de la línea de borde del área de influencia.
	colorRelleno_eben	text	Yes	Es el color que tendrá el área de influencia.
	opacidadRelleno_eben	text	Yes	Es la intensidad de color que tendrá el área de influencia.
	eliminado_eben	boolean	Yes	Almacenará un true o false

Nota. Diccionario de datos de la tabla tb_ estilobeneficiario

Elaborado por: Cofre Víctor & Toledo Stalin

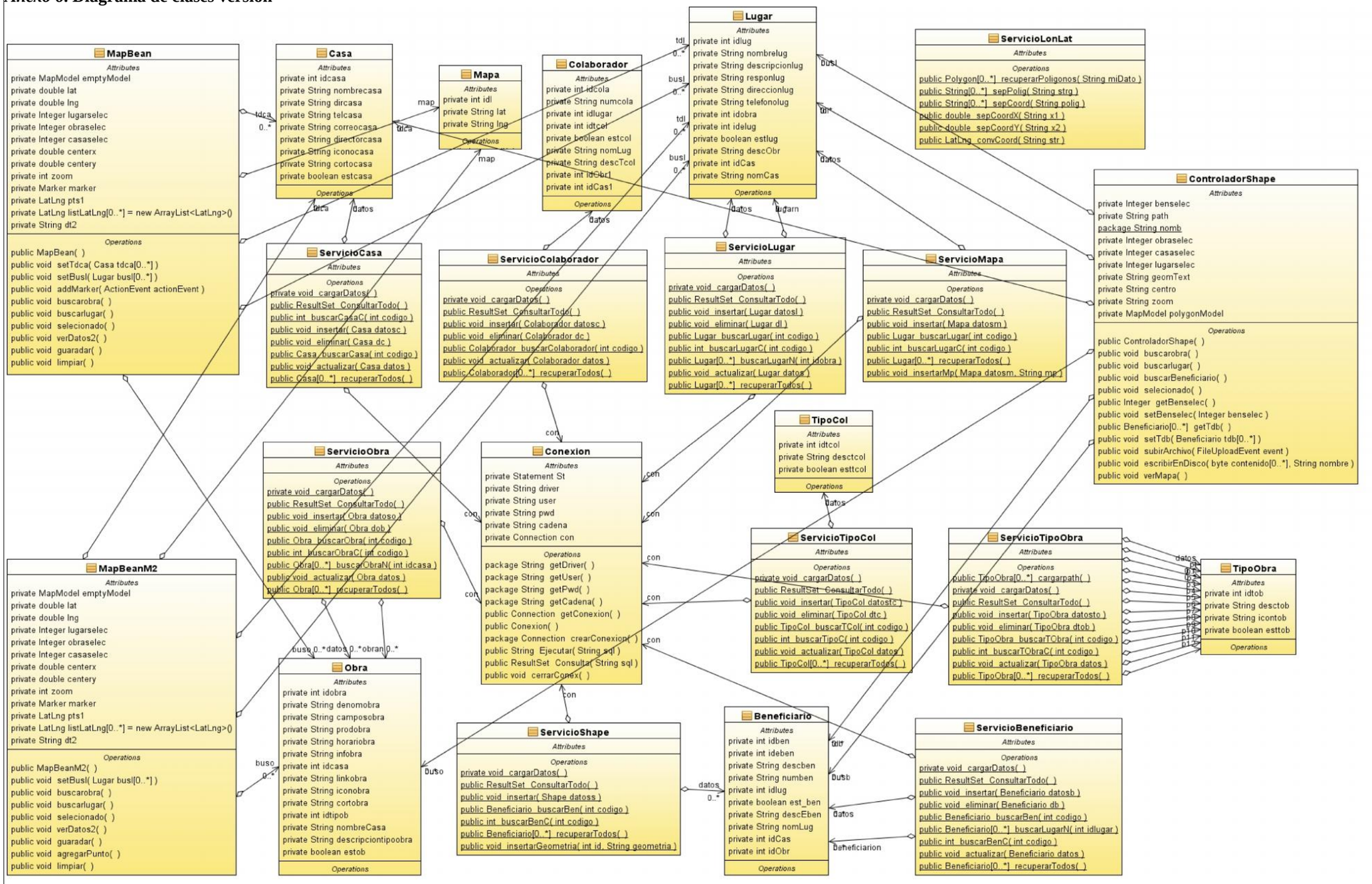
Diccionario de datos tabla foto Lugar

Nombre	tb_fotolugar			
Descripción	Almacena la información de fotografías que se asignara al lugar			
Primary Key	id_flug			
Key	**Column Name**	**Data Type**	**Not Null**	**Description**
PK	id_flug	serial	Yes	Identificación única de la foto que se le asigna a un lugar
FK	id_lug	integer	Yes	Representa al indicador del lugar
	descripcion_flug	text	Yes	Es la descripción o nombre de una foto.
	pathfoto_flug	text	Yes	Se especifica el directorio en el cual se guardaran las fotos del lugar.
	estado_flug	boolean	No	Almacenará un true o false

Nota. Diccionario de datos de la tabla tb_ fotolugar

Elaborado por: Cofre Víctor & Toledo Stalin

***Anexo 6*. Diagrama de clases versión**



Elaborado por**:** Cofre Víctor & Toledo Stalin