Desarrollo de un Servicio de Nomenclátor en Web

Eduardo Barrios Garrido
Departamento de Ingeniería Informática y Ciencias de la Computación
Universidad de Concepción

Profesor Guía: Dr. Diego Seco Naveiras

Informe de Memoria para optar al título de
INGENIERO CIVIL INFORMÁTICO

Concepción, Chile
Marzo, 2019

Resumen

Las Infraestructuras de Datos Espaciales (IDEs) son un conjunto de tecnologías y estándares que facilitan la transferencia de Información Geográfica (IG) entre organizaciones de diferente tipo. Uno de sus componentes principales, el servicio de Nomenclátor, es un catálogo de entidades que permite obtener información descriptiva de carácter geográfico de dichas entidades reales.

Dada la importancia de la información geográfica en la toma de decisiones a nivel público y privado, y la falta de iniciativas concretas por parte de IDE Chile que contemplen el desarrollo de este servicio, se desarrolla un Servicio de Nomenclátor en Web que permite consultas mediante HTTP de entidades y visualización de su información por medio de una interfaz gráfica que incluye funcionalidades de creación de una nueva entidad y edición de las existentes, además de visualización de información en XML y JSON.

Índice General

Capítulo 1

Introducción

El reconocimiento de la importancia de la información geográfica y de medios que la provean de manera ordenada y descriptiva de tal manera de que sea de valor en la toma de decisiones en diversos niveles de organismos públicos o privados se produjo en la conferencia de las Naciones Unidas sobre el medio ambiente en Río de Janeiro de 1992 [17]. Entre todos los temas debatidos como la protección ambiental, cooperación internacional y desarrollo, uno clave que marca el principio de lo que sería el desarrollo de las Infraestructuras de Datos Espaciales (IDEs) es que para tratar los problemas debatidos era necesario mecanismos para aprovechar al máximo la información geográfica, ya que gracias a ella se podría tomar decisiones relevantes. Fue claro entonces que era necesario coordinarse en el acceso, distribución, formas y empleo de la información geográfica.

En particular, como fruto de la reunión de las Naciones Unidas se adoptó el Programa 21, el cual define en su Principio 10, la importancia de otorgar a todas las personas acceso adecuado a la información sobre el medio ambiente que dispongan las autoridades públicas, con el objetivo de incluir a los ciudadanos al momento de tratar las problemáticas ambientales. Este principio es clave, ya que para llevarlo a cabo deben existir mecanismos que pongan a disposición general información geográfica, ya que esta es fundamental para realizar cualquier estudio del medio [17].

A raíz de lo anterior diversos organismos públicos y privados crean mecanismos para utilizar la información geográfica para sus fines, lo cual produjo, entre otras, diferencias en la representacion, almacenamiento y distribución de la información. No es hasta que se superaron los límites y restricciones que regularon el uso de las Sistemas de Información Geográfica (SIG), que se logró definir normas y estándares comunes con respecto al tema. Lo último gatilló el desarrollo de las IDEs como un medio que aprovecha al máximo la información geográfica.

Alrededor del mundo la mayoría de los países, incluyendo algunos de sus niveles administrativos menores tiene sus propias IDEs, otorgándoles al día de hoy ventajas que ya no se limitan a problemas sobre el medio ambiente, ya que otorgan información importante en otras áreas como la seguridad nacional, recreacionales, planificación urbanística, etc.

Las IDEs, al día de hoy, aunque están compuestas principalmente por un servicio de catálogo de metadatos, un servicio de publicación de mapas y un servicio de Nomenclátor, no

todas ellas poseen todos estos servicios, la mayoría de ellas poseen catálogo de metadatos y publicación de mapas. Esta última en general viene de la mano con un medio de visualización.

En el caso particular de Chile, la distribución pública de los servicios de la IDE está cargo del Ministerio de Bienes Nacionales y de su subdivisión IDE Chile, esta última otorga una plataforma Web en la cual se puede acceder a los servicios de metadatos y publicación de mapas. Además posee de un visor que permite observar los servicios mencionados. Por ende, es la falta del servicio de Nomenclátor lo que motiva el desarrollo de esta Memoria de Título.

1.1. Objetivo General

El objetivo de esta memoria es diseñar e implementar un servicio de nomenclátor bajo estándares y uso de Software de código libre, que permita realizar consultas mediante HTTP y una interfaz pública, esta última debe permitir actualizar y crear nueva información.

1.2. Objetivos Específicos

Los objetivos específicos de esta memoria son:

- Diseñar un modelo de servicio de Nomenclátor que permita el almacenamiento y gestión de los nombres geográficos o topónimos en una base de datos relacional cumpliendo las especificaciones WFS-G.

- Desarrollar un servicio web que acepte peticiones HTTP y devuelva información en formato XML y JSON.

- Desarrollar una interfaz gráfica amigable que permita realizar diversos tipos de consultas.

- Agregar funciones al servicio Web que permita agregar nueva información y editar la existente.

- Agregar funciones al servicio Web que permita visualizar la información descriptiva y geográfica.

Capítulo 2

Conceptos previos y trabajo relacionado

A continuación se definen conceptos necesarios asociados a este trabajo para una mejor comprensión.

Los nombres geográficos también conocidos como topónimos por definición, son un "conjunto de nombres propios de lugar de un país o de una región", estos permiten designar cada entidad geográfica de manera individual e independiente del resto. Una entidad es una abstracción del mundo real asociada a una ubicación concreta. En otras palabras un topónimo es el nombre propio dado a un accidente o evento sobre la superficie de la tierra que corresponde a una zona específica, esta puede ser una palabra, una combinación de palabras o una expresión concreta [5].

La importancia de la toponimia y su utilización normalizada, recae en el hecho de identificar plenamente y sin ambigüedades un lugar, ayudando así a un ordenamiento del conocimiento geográfico dando información descriptiva. Además proporciona un conjunto de datos para las IDEs.

Una Infraestructura de Datos Espaciales (IDEs) es un conjunto de datos espaciales, tecnologías, políticas y estándares cuyo objetivo es facilitar y proveer la utilización de información geográfica, ayudando así al desarrollo social, económico y ambiental [4]. Está compuesto principalmente por un servicio de catálogo de metadatos, un servicio de publicación de mapas y un servicio de Nomenclátor, aunque también puede incluir otros servicios dependiendo del dominio.

En el ámbito geográfico un servicio de Nomenclátor está compuesto por un catálogo de entidades con información descriptiva de ellas. Su objetivo es entregar una o más entidades en respuesta de una consulta la cual habitualmente consiste en algún identificador de la entidad que se busca, la cual puede ser el nombre, tipo de entidad y localización geográfica. Open Geospatial Consortium (OGC) define modelos de datos y funcionalidades que debe cumplir un servicio de Nomenclátor bajo el perfil Gazetteer Service WFS-G.

2.1. Trabajo relacionado

Debido a la importancia de las IDEs establecida, en un principio, en la Reunión de las Naciones Unidas en Río de Janeiro en 1992 [17], existen diversas IDEs a nivel mundial. Aquí se mencionan algunas.

En el contexto internacional se debe mencionar a la Global Spatial Data Infrastructure Association (GSDI), organización que agrupa a otras organizaciones y agencias alrededor del mundo con el objetivo de promover el desarrollo de las IDEs de forma coordinada de manera de cubrir todo el mundo.

Se debe destacar a Geonames.org, la cual es una base de datos geográfica gratuita, de la cual se obtuvo la información para las base de datos que se utiliza en este trabajo. Algunos de los servicios que provee Geonames es la consulta de nombres geográficos por URL que arroja la información solicitada en formato XML y JSON. Además posee una interfaz gráfica la cual permite visualizar la información de nombres geográficos buscados, editar información y agregar nuevas entidades. Los servicio que provee Geonames están bajo estándares y resulta el más completo de todos.

En España las IDEs están en un nivel alto de desarrollo, teniendo IDEs regionales y locales desarrollados bajo el estándar WFS-G y el Modelo de Nomenclátor de España (MNE). Estas IDEs funcionan como nodos de una amplia red que trabaja en conjunto. Dentro de ellas mencionaremos la de Aragón y Galicia.

El servicio de Nomenclátor entregado por el Gobierno de Aragón permite, mediante una interfaz gráfica, realizar búsquedas por el nombre de la entidad, dando la opción de que los resultados coincidan con todas las palabras ingresadas, algunas de las palabras, con todas las palabras en el orden que se escribieron, coincidencia exacta y que contenga lo buscado. También se puede agregar la municipalidad dentro de la cual se desea realizar la búsqueda. El resultado entregado consiste en el nombre de la entidad, el municipio a la cual pertenece, el tipo de entidad y sus coordenadas geográficas. Da la opción de editar las entidades previo inicio de sesión dentro del sistema. No permite La creación de nuevas entidades.

El servicio de Galicia posee más funcionalidades que el anterior, éste permite realizar búsquedas que permiten definir los niveles administrativos de Provincia, Ayuntamiento y Parroquia o Municipalidad. Además permite establecer el tipo de topónimo que se desea encontrar. La posición geográfica de los resultados pueden ser exportadas en formato PDF, KML, GPX y CSV. No permite edición, ni creación de entidades.

En el contexto nacional, no existe iniciativas concretas de desarrollo de un Servicio de Nomenclátor por parte de IDE Chile. Sin embargo, este posee un servicio de catálogo de

metadatos y servicio de publicación de mapas, además de un visor de mapas, como en la mayoría de los Sistemas IDEs.

2.2. Tecnologías usadas

Para la selección de las tecnologías utilizadas se consideraron aspectos como la documentación disponible, conocimiento previo, código libre y buenas prácticas de desarrollo de Software. A continuación se detallarán.

2.2.1. Base de datos

Se utilizó el sistema de gestión de base de datos PostgreSQL, desarrollado por la comunidad PostgreSQL Global Development Group en lenguaje C, debido a que es un motor de base de datos relacional orientada a objetos, de código abierto, multiplataforma, capaz de manejar grandes volúmenes de información de manera concurrente, está disponible para Linux, Solaris, FreeBSD, Mac OSX y Windows y es soportado por diversos Frameworks. Ademas posee la extensión PostGIS, que añade soporte de objetos geográficos y permite consultas SQL espaciales necesarias para el desarrollo de este trabajo. Dentro de las opciones de motores de bases de código abierto barajados para el desarrollo del sistema, PostgreSQL es considerado el más avanzado con más de 30 años de desarrollo y su extensión PostGIS son ampliamente utilizados [19].

2.2.2. Framework

El sistema fue desarrollado sobre Spring Framework, plataforma Java de código abierto, escrito por Rod Johnson bajo la licencia Apache 2.0 en junio de 2003. Este Framework promueve las buenas prácticas de programación al estar basado en Plain Old Java Object (POJO), estos son una instancia de una clase, la cual no implementa ni extiende métodos de otras clases. Un ejemplo simple sería una clase con solo sus atributos y métodos. En términos generales y como ejemplo, en el sistema desarrollado son utilizados para expresar cada entidad del modelo de datos y contenedores de información enviada desde la vista al controlador y del modelo a la vista.

El concepto de POJO es importante en el desarrollo de aplicaciones Java complejas, ya que las clases deben ser lo más independientes posible para reutilizarlas y testearlas de manera unitarias. Para lograr esto y a la vez poder usar las clases entre ellas se utiliza el patrón de diseño orientado a objetos llamado Inyección de Dependencias o DI, éste es implementado mediante la Inversión de Control (IoC).

En la programación estructurada habitual el flujo de los procesos está definida por el programador, en cambio, en la IoC se definen comportamientos del sistema y se deja que el Framework determine el flujo de las acciones.

La DI es un evento concreto de la IoC. La IoC inyecta una clase dentro de otra de la cual depende, en lugar de instanciar una clase dentro de otra.

En la práctica los conceptos de DI y IoC son llevados a cabo por medio de un contenedor IoC de Spring, el cual crea objetos y los administra. Estos objetos son conocidos como Beans.

Existen dos maneras para configurar la DI en Spring Framework. La manera tradicional es usando un archivo XML donde se declaran los distintos Beans para el contenedor IoC. Otra manera es por medio de anotaciones directas en las clases o métodos del código.

Un concepto importante de Spring Framework es el marco de Programación Orientada a Aspectos (AOP). Existen funciones comunes que son transversales dentro de una aplicación y AOP permite separar estas funcionalidades de las demás lógicas del sistema, algunos ejemplos son la seguridad, conexiones a servidores, propiedades y almacenamiento de caché.

Spring Framework posee diversos módulos que pueden ser utilizados dependiendo de la aplicación que se desea desarrollar. En la Figura 2.1 se aprecia la arquitectura de Spring. Se hablará de los módulos pertinentes a lo desarrollado en este proyecto [8].

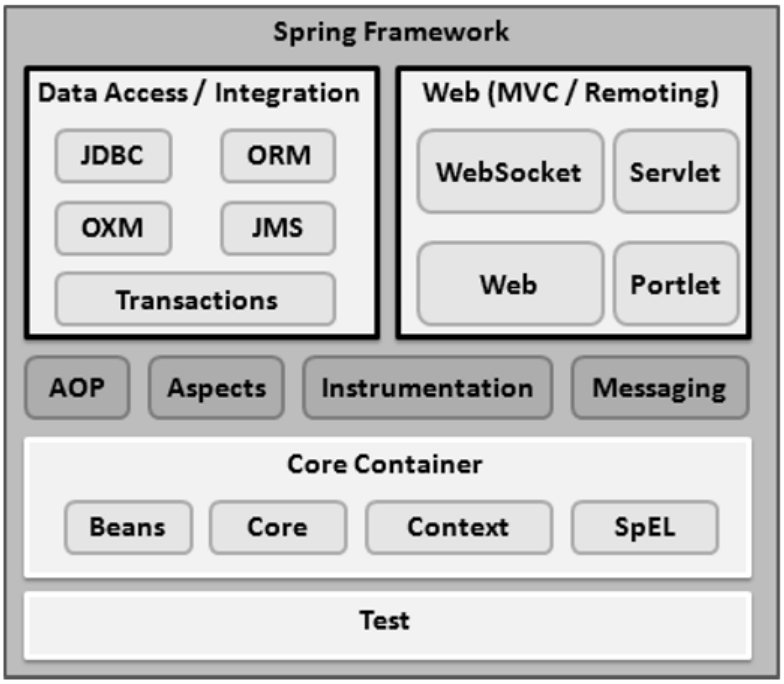


Figura 2.1: Arquitectura de Spring Framework [20].

- Core proporciona las características de IoC y de inyección de dependencia.
- Bean proporciona BeanFactory, encargado de instanciar los distintos beans.
- AOP le entrega al Framework la capacidad de programación orientada a aspectos.
- El módulo ORM proporciona la capacidad de integrar APIs de mapeo relacional de objetos, incluidas JPA, JDO, Hibernate e iBatis.
- Transacción admite la gestión programática y declarativa de transacciones para clases que implementan interfaces especiales y para todos sus POJO.
- Web-MVC es el Framework Spring MVC para desarrollar aplicaciones web con el patrón MVC.
- Test provee de soporte para el testing de la aplicación por medio de los marcos JUnit o TestNG.

Otro concepto utilizado en el desarrollo de este proyecto es el de Objeto de Acceso a Datos (DAO), el cual se refiere a las interacciones con la base de datos.

Los DAOs proveen de funcionalidades de lectura y escritura a la base de datos por medio de interfaces y facilitan el uso de tecnologías como Hibernate y JPA.

Los DAOs están compuestos por una interfaz que define los métodos que se realizan, una clase que las implementa y es la encargada de obtener los datos y una clase POJO que representa la entidad de la base de datos con sus atributos y métodos de obtención y configuración, que además sirve como contenedor de la información obtenida de la base de datos [16].

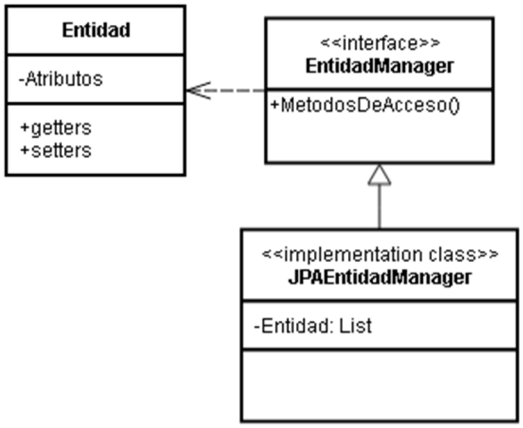


Figura 2.2: Patrón DAO utilizado en el sistema [21].

Hibernate es un Software de código abierto desarrollado por Red Hat, lanzado el año 2001 bajo la licencia GNU LGPL. Esta herramienta provee de mapeo objeto-relacional para la plataforma Java y facilita la relación de atributos de la base de datos con los objetos de la aplicación por medio de XML o anotaciones en los beans [10].

JPA o Java Persistence API, es una API de persistencia para la plataforma Java desarrollada por Sun Microsystems, tiene como principal objetivo el uso de POJOs al interactuar con la base de datos [11].

En particular para el desarrollo se utilizó Spring Tool Suite (STS), un entorno de desarrollo listo para usar, basado en Eclipse para aplicaciones Spring Framework, este incluye componentes necesarios que nos facilitan la configuración de aspectos del sistema como Maven, Tomcat y JUnit.

Maven es una herramienta creada por Jason Van Zyl en 2002 y ahora desarrollada por Apache Software Fundation, bajo licencia Apache 2.0 y programado en Java. Nos permite gestionar y construir el proyecto en formato XML. Está encargado de gestionar las librerías y módulos necesarios para el funcionamiento del sistema [12].

Tomcat es un contenedor de servlets desarrollado por Apache Software, lanzado en 1999 bajo licencia Apache 2.0 y programado en Java. Su principal funcionalidad es operar como servidor Web [18].

Junit es un Framework de código abierto para la plataforma Java que permite automatizar pruebas unitarias y evaluar el correcto comportamiento del código, fue desarrollado por Erich Gamma y Kent Beck [13].

Dentro de STS el desarrollo utilizó el marco Spring Web MVC, para proporcionarnos la arquitectura MVC. El patrón MVC divide las lógicas del sistema dejando al Modelo encargado de las interacciones con la base de datos, a la Vista de la información de cara al usuario y al Controlador de administrar las solicitudes del usuario al modelo y entregarlas a la Vista. El marco Spring Web MVC posee una arquitectura propia que se puede apreciar en la Figura 2.3

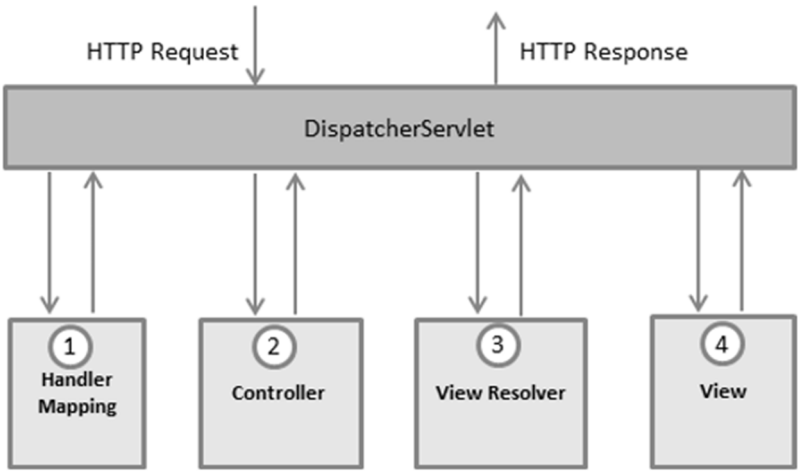


Figura 2.3: Arquitectura de Spring Web MVC [15].

Spring Web MVC utiliza una la clase DispatcherServlet para manejar las solicitudes y respuestas HTTP. Esta clase llama al controlador que resuelve la petición HTTP mediante HandlerMapping. Luego el controlador llama a los servicios encargados de elaborar la respuesta a la solicitud según el método POST o GET definido en el método del controlador. Después DispatcherServlet recibe el nombre de la vista a la cual debe pasar los datos entregados por el modelo y se ayuda de ViewResolver para seleccionarla [15].

2.2.3. Librería para datos espaciales en Web

Para mostrar la información geográfica se utiliza la biblioteca de código abierto Leaflet escrita en JavaScript. Esta es compatible con la mayoría de las plataformas móviles y de escritorio. Las principales características que determinaron su elección

fueron la poca complejidad de uso, estar bien documentada, ser más ligero y de mejor rendimiento. Además permite manejo de archivos GeoJSON [14].

2.2.4. Otros lenguajes de programación

Las vistas fueron programadas en archivos JSP, tecnología que ayuda a crear páginas Web dinámicas basadas en HTML. Para su diseño se utiliza el CSS de Bootstrap.

Para manejar los eventos de las vistas e información desde la vista al controlador y viceversa se utiliza JavaScript.

La información resultante de las búsquedas es mostrada en una tabla configurada gracias al plugin de JavaScript DataTables que agrega varias funcionalidades, entre ellas la paginación, ordenar columnas y búsqueda dentro de la tabla.

Capítulo 3

Sistema Desarrollado

El sistema desarrollado consta de dos partes. Un sistema Web con interfaz gráfica para la consulta de nombres geográficos acotado a distintos tipos de búsqueda, pudiendo especificar las distintas localidades o niveles administrativos en donde realizarla, además de permitir enmarcar los resultados que se deseen dentro del cuadro delimitado por el visor de mapa. Esta interfaz Web permite también la creación y edición de entidades.

La segunda parte del sistema entrega resultados en formato XML o JSON de entidades por medio de consultas por URL.

A continuación se da a conocer el proceso de desarrollo del proyecto de esta memoria.

3.1. Metodología

Se basó en la metodología de desarrollo de Software para la elaboración de este trabajo, con el fin de que fuera estructurada, planificada y controlada.

En un principio se analizaron los requerimientos del sistema para entender y describir su comportamiento y funcionalidades, una vez aclarado los alcances del sistema, se definieron los casos de uso. Luego se realizó una revisión bibliográfica sobre el tema del proyecto, donde se tomó conciencia de su importancia y desarrollos a nivel internacional. Además se recopiló información útil para el desarrollo del proyecto, en específico, la información a emplear en la base de datos.

Posteriormente se barajaron distintas herramientas a utilizar para el desarrollo del proyecto, en particular con respecto al Framework y sus componentes. En su momento se alcanzó a elaborar un primer prototipo en base al Framework Codeigniter, debido a una experiencia y conocimiento previo. Este fue desechado en pro de un sistema más ordenado, fácil de mantener, seguro y fundado en buenas prácticas de desarrollo de Software.

Al escoger Spring Framework, por ser una mejor opción para la implementación del sistema, se enfrentó el hecho de aprender a utilizar una nueva herramienta.

Durante el diseño se elaboró la arquitectura del sistema, el modelo de la base de datos y el diagrama de casos de uso. Debido a que la implementación está basada en Spring Framework el patrón arquitectónico del sistema es el Modelo-Vista-Controlador o MVC, el

cual se caracteriza por dividir las interacciones dentro del sistema en tres partes; el Modelo encargado de interactuar con la base de datos, el Controlador encargado de la implementación de la lógica de negocio de la aplicación y la Vista que se encarga de entregar la información al usuario.

Por último, para la implementación del proyecto se utilizó una metodología iterativa incremental guiada por el desarrollo de las funcionalidades o casos de uso, donde en cada iteración se implementa una funcionalidad y se corrigen errores hasta solidificar la funcionalidad realizada en la iteración anterior.

3.2. Análisis

En un principio se establecieron las especificaciones de los requerimientos del sistema y se tomó conciencia de su comportamiento y funcionalidades. Para el sistema Web se definen los siguientes requerimientos funcionales:

- Realizar consultas mediante peticiones HTTP.
- Visualizar la información entregada en formato XML o JSON.
- Devolver una o más entidades en respuesta a una consulta:
    - Coincidencia exacta con la palabra consultada.
    - Que comience por la palabra consultada.
    - Contenga cualquiera de las palabras consultadas.
- Filtrar la consulta por criterios de área a consultar.
- Desplegar información de cada entidad devuelta en la consulta:
    - Identificador geográfico.
    - Nombre.
    - Nombres alternativos.
    - Coordenadas Geográficas: Latitud, Longitud.
    - Clase identidad.
    - Código de característica.
    - Niveles de administración: Región, provincia, comuna.
- Visualizar geográficamente las entidades.
- Centrar la vista geográfica en la respuesta seleccionada.
- Crear una nueva Entidad con los siguientes atributos:
    - Coordenadas Geográficas.
    - Nombre.
    - Niveles Administrativos.
    - Clase identidad.
    - Código de característica.
    - Nombres alternativos.
    - Zona Horaria.

- Eliminar una entidad.
- Editar la siguiente información de las entidades:
    - Coordenadas geográfica.
    - Nombre.
    - Clase identidad.
    - Código de característica.
    - Zona Horaria.
    - Niveles administrativos.
    - Nombres alternativos.

Los requerimientos no funcionales son:

- Uso de Software libre.
- Interfaz gráfica sencilla.

3.3. Casos de Uso

Teniendo claro los requerimientos funcionales se definieron los casos de uso para modelar los procesos que realiza el usuario con sistema, estos se pueden apreciar en el diagrama de casos de uso de la Figura 3.1. Se consideró como actor a un usuario sin restricciones.

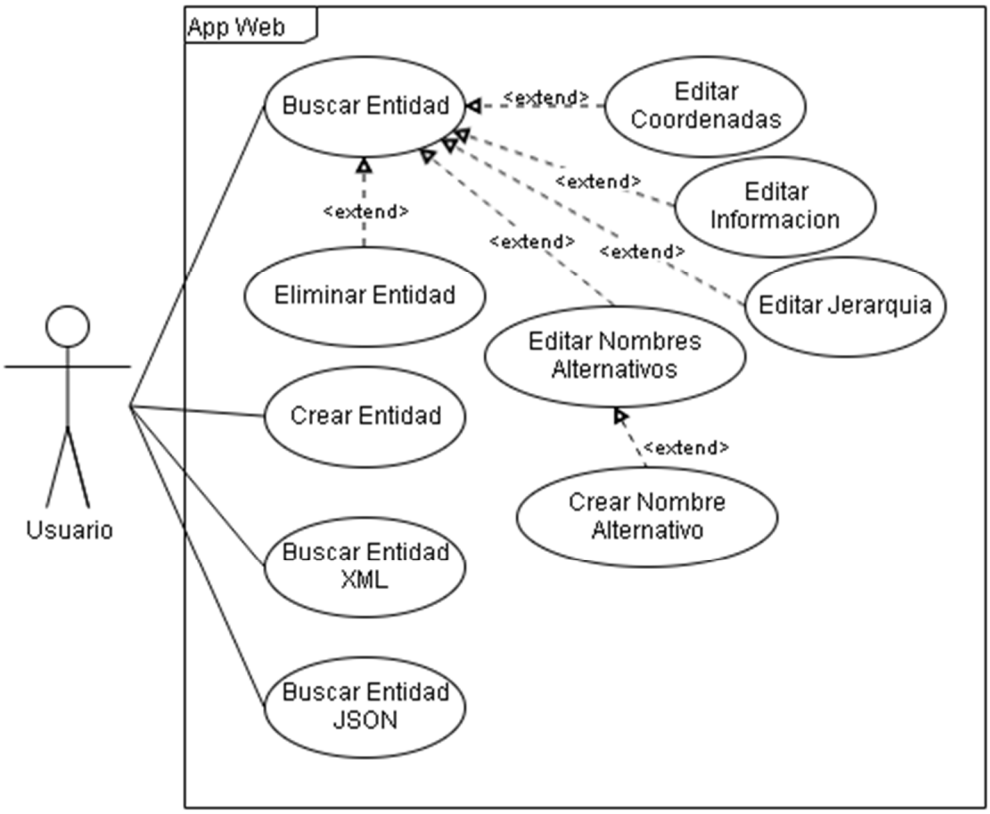


Figura 3.1: Casos de Uso.

Dentro del sistema se consideran tres casos de uso importantes; Buscar Entidad, Crear Entidad y Buscar Entidad XML o JSON.

El caso de uso Buscar Entidad consiste en ingresar un nombre geográfico y encontrar entidades que coincidan con lo ingresado. Existen tres tipos de búsqueda de coincidencia con respecto al nombre geográfico; coincidencia exacta, comienza con la palabra consultada y que contenga cualquiera de las palabras. En esta última las palabras deben ser separadas por comas. Además la búsqueda puede o no, ser acotada de 2 formas incluyentes, ya sea delimitando los resultados a la región mostrada por el visor de mapa y/o por uno o todos los niveles administrativos que existen en Chile; Región, Provincia, Comuna. Después de realizar la búsqueda y obtener resultados, estos pueden ser eliminados o editados, estas funcionalidades son cubiertas por los siguientes casos de uso:

- **Editar Coordenadas:** El usuario puede ingresar a cambiar la posición geográfica de cualquiera de las entidades entregadas como resultado de la búsqueda, en específico, se modifican sus coordenadas de latitud y longitud. Luego de la modificación se restablece el marcador en la nueva posición dentro del visor de mapa.

- **Editar Información:** El usuario puede modificar la información general de cada entidad. Los atributos que pueden ser modificados en este caso de uso son:

    - Nombre.
    - Clase identidad.
    - Código de característica.
    - Zona Horaria.

- **Editar Jerarquía:** Este caso de uso permite modificar cualquiera de los tres niveles administrativos de cada entidad: Región, Provincia y Comuna.

- **Editar Nombres Alternativos:** Se puede ingresar a la lista de nombres alternativos de cada entidad. Dentro de esta lista el usuario puede modificar el nombre y el código de lenguaje de cada una de ellas. Además, en este punto el usuario puede llevar a cabo el caso de uso Crear Nombre Alternativo.

    - **Crear Nombre Alternativo:** El usuario tiene como opción agregar otro nombre alternativo a la lista de nombres alternativos de la entidad.

- **Eliminar Entidad:** El usuario puede eliminar cualquier entidad de los resultados de la búsqueda.

El usuario puede ingresar al caso de uso Crear Entidad al seleccionar con el botón derecho del ratón un punto geográfico dentro del visor de mapa, esto despliega la opción de Crear nueva Entidad. Dentro del caso de uso el usuario puede modificar las coordenadas de la nueva entidad e ingresar su información. Los atributos que pueden ser ingresado son:

- Coordenadas geográficas: Latitud, Longitud.
- Nombre.
- Niveles administrativos: Región, Provincia, Comuna.
- Clase identidad.
- Código de característica.
- Nombres Alternativos: Estos deben ser separados por coma.
- Zona Horaria.

Los casos de uso Buscar Entidad XML o JSON, son consultas por medio de la URL con características similares al caso de uso Buscar Entidad. Para acceder a estas funcionalidades se debe ingresar a las URLs dependiendo del tipo de formato de la respuesta que se desea. En el caso particular se puede ingresar por las siguientes URLs:

- http://localhost:8080/springapp/busquedaxml?
- http://localhost:8080/springapp/busquedajson?

Para realizar la búsqueda se deben ingresar los siguientes parámetros separados por el caracter &:

- **nombre_igual:** Busqueda de entidades con coincidencia exacta.
- **nombre_partePor:** Búsqueda de entidades que comiencen con el nombre geográfico.
- **nombre_contCual:** Búsqueda de entidades que contengan cualquiera de las palabras. Cada una de las palabras deben ser definidas de esta manera: nombre_contCual=nombregeografico&nombre_contCual=nombregeografico2&...
- **admin1:** Acota la búsqueda al primer nivel administrativo, Región. Se debe ingresar el código del nivel.
- **admin2:** Acota la búsqueda al segundo nivel administrativo, Provincia. Se debe ingresar el código del nivel.
- **admin3:** Acota la búsqueda al tercer nivel administrativo, Comuna. Se debe ingresar el código del nivel.
- **norte, sur, este, oeste:** Acota la búsqueda a un cuadro delimitado por estos parámetros. Deben ser usados los 4 a la vez.

Para llevar ejecutar estas funcionalidades es necesario incluir uno de los tipos de búsqueda: nombre_igual, nombre_igual, nombre_contCual. Por defecto la búsqueda se realiza en todo el territorio. Los demas parámetros tienen la finalidad de acotar el marco de búsqueda. Se debe aclarar que en el caso de los niveles administrativos, sólo es necesario ingresar uno de ellos.

A continuación algunos ejemplos de uso:

Ejemplo 1:
http://localhost:8080/springapp/busquedaxml?nombre_igual=Concepción&admin1=06

Ejemplo 2:
http://localhost:8080/springapp/busquedaxml?nombre_partePor=Cerro&admin3=06101&norte=-33.89&sur=-34.38&este=-70.31&oeste=-71.03

3.4. Diseño

Desde un principio se determinó que la arquitectura del Software debía seguir el patrón Modelos-Vista-Controlador (MVC), ya que está demostrado que es un patrón de diseño que provee de buenas prácticas de desarrollo, debido a que separa el sistema en 3 componentes, cada una con tareas específicas.

El Modelo se encarga de las interacciones con la base de datos.

La Vista es la interfaz del sistema y se encarga de entregar la información al usuario obtenida a partir del modelo.

El Controlador administra las solicitudes de la vista hacia el modelo y las respuestas de este último hacia la vista, implementado así la lógica de negocio de los casos de uso.

Alguna de las ventajas del patrón MVC son: permite que el sistema escale de forma modular, ayuda a la mantención y corrección de errores, reutilización de código y ordena el código lo que permite controlar el comportamiento del sistema.

A continuación en la Figura 3.2, se puede apreciar la arquitectura del sistema. En este diagrama SGBD es el servicio gestor de base de datos, que como se mencionó anteriormente se utilizó PostgreSQL y su extensión PostGIS.

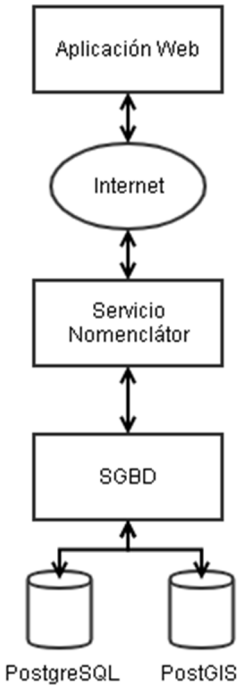


Figura 3.2: Arquitectura del Sistema.

3.5. Modelo de Datos

En la figura 3.3 se puede apreciar el modelo de datos que se elaboró para el sistema. Este es un Modelo Entidad-Relación donde la entidad principal es Geoname, ya que en ella se guardan los nombres geográficos con la mayoría de sus atributos, como la posición geográfica representada con los atributos de latitud, longitud y punto[1], el nombre y el código de sus niveles administrativos, entre otra información. Otras tablas importantes son las entidades ADMIN1, ADMIN2 y ADMIN3, estas corresponden a las Regiones, Provincias y Comunas respectivamente, no hay una relación entre ellas, es la tabla herencia la cual asocia los elementos de la tabla Geoname que sean Comunas con su respectiva Provincia, y ésta a su vez, con la Región a la cual pertenece. También vale la pena mencionar la entidad NombreAlternativo, la cual se relaciona con Geoname para almacenar los diversos nombres alternativos de cada nombre geográfico. Estos nombres alternativos pueden estar escritos en diferentes lenguajes, por lo que la entidad posee un atributo que asocia el nombre alternativo con la entidad IsoLenguaje para así determinar a cuál lenguaje pertenece. La entidad ClaseIdentidad guarda los tipos de nombres geográficos que existen, estos son:

- A: país, estado, región, ...
- H: arroyo, lago, ...
- L: parques, zona, ...
- P: ciudad, pueblo, ...
- R: carretera, ferrocarril
- S: lugar, edificio, granja
- T: montaña, colina, roca, ...
- U: submarino
- V: bosque, brezo, ...

Cada uno de estos tipos tiene asociado diversos códigos de características los cuales están representados en la entidad CodigoCaracteristica. En Esta última se representan subcategorías de los tipos de la entidad ClaseIdentidad. Por ejemplo, el tipo L que agrupa los Geonames en parques, zona, etc., está compuesto por códigos que representan categorías más pequeñas del tipo L que son: AGRC; colonia agrícola, Amus; parque de atracciones, BSND; cuenca de drenaje, entre otros [22].

[1] Representación geométrica tipo punto de longitud y latitud utilizada para realizar consultas espaciales gracias a la extensión PostGIS.

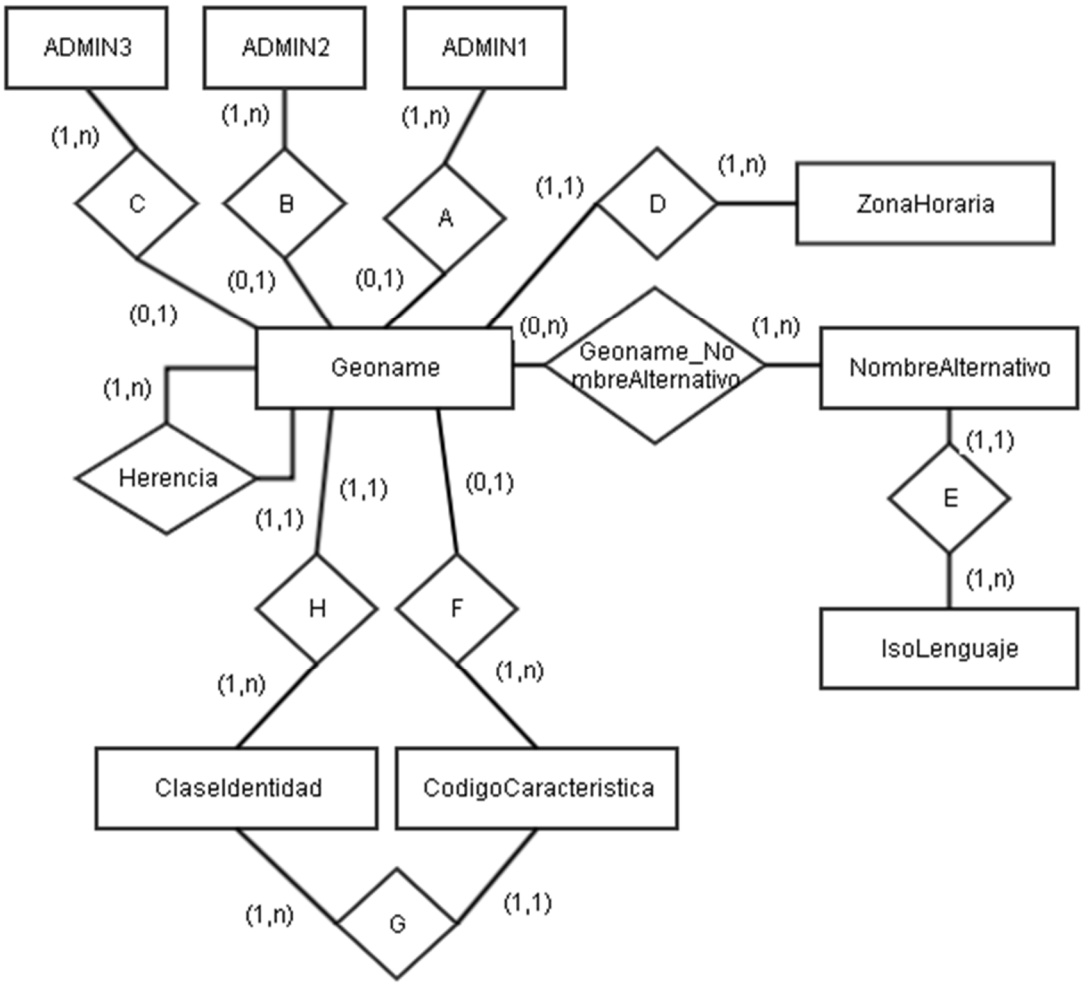


Figura 3.3: Modelo Entidad-Relación.

El Apéndice B muestra el modelo relacional y se pueden apreciar los atributos de las tablas. Se debe mencionar que tanto para modelar como poblar la base de datos fue fundamental la información que entrega de forma gratuita Geonames.org [6].

3.6. Implementación

La implementación del sistema se llevó a cabo en el entorno de desarrollo Spring Tool Suite (STS), el cual está basado en Eclipse y personalizado para elaborar aplicaciones Spring listo para usar. Viene integrado con Tomcat, Maven y JUnit, entre otros [9]. Tomcat es un contenedor de servlets y es necesario para que opere como servidor Web, Maven es una herramienta que nos permite gestionar y construir el proyecto agregando las librerías, módulos y dependencias necesarias para el funcionamiento del sistema y JUnit es un Framework que nos permite automatizar pruebas en el sistema.

La Inyección de Dependencias (DI) fue configurada por medio de anotaciones, declarando tags que provee Spring en las clases y/o métodos definiendo sus comportamientos de manera automática por el Framework. Esto último es la principal ventaja en comparación a una configuración por medio de un fichero XML, donde se deberían registrar los Beans a utilizar en el sistema. Si el sistema es complejo o extenso, el fichero XML crece con respecto a los beans y se hace poco legible, aun así, existe la opción de una configuración mixta.

3.6.1. Configuración del entorno

Dentro del proyecto se crean 3 carpetas fuentes para estructurar los archivos de Java del proyecto. Uno de ellos está destinado a almacenar los distintos archivos con los cuales se realizarán las pruebas del sistema, otra carpeta está destinada para las distintas propiedades y la última para almacenar las distintos aspectos del sistema, como por ejemplo:

- DataSource: Configuración de acceso a la base de datos.
- EntityManagerFactory: Gestiona las entidades.
- DispatcherServlet: Interactúa con el controlador y la vista.

La última carpeta fuente también está subdividida en carpetas para almacenar los archivos utilizados para las capas de Controlador y Modelo del patrón MVC. La capa de Modelo está representada por DAOs, este patrón se divide en 3 clases, la clase que representa la entidad del modelo de datos con sus atributos y métodos para obtenerlos y establecerlos, otra clase de interfaz y una clase que implementa los métodos de la interfaz. Esta última clase llama a una cuarta clase que contiene las operaciones a la base de datos.

Una vez organizados los contenedores para los diversos aspectos del sistema se deben configurar las dependencias del sistema, para llevar a cabo esto utilizamos Maven por medio del archivo pom.xml donde se establecen las dependencias a usar.

3.6.2. Controladores

En concreto en el sistema se implementaron 3 controladores, uno para administrar las solicitudes de la interfaz gráfica y otros 2 para las funcionalidades de búsqueda con respuesta en formato XML o JSON. La manera en que actúa el controlador con la interfaz gráfica es mediante Javascript y AJAX con peticiones tipo POST y recibiendo la información entregada por el controlar como texto JSON. Por otra parte los controladores que devuelven información en formato XML y JSON poseen métodos que operan con solicitudes de tipo GET.

Los controladores interactúan con la capa de Modelo por medio de la clase interfaz del patrón Objeto de Acceso a Datos (DAO) del cual desea obtener información.

3.6.3. Objetos de Acceso a Datos (DAOs) y Repositorios

El patrón DAO nos permite desacoplar aún más la capa de Modelo del sistema, agregando mayor independencia y separando la tecnología utilizada para la persistencia de datos de la aplicación. En el caso del sistema desarrollado se utiliza Java Persistence API (JPA) para la persistencia de datos. Se debe mencionar que se utiliza Hibernate como proveedor de JPA, esta es configurada como un aspecto dentro del sistema. Hibernate nos provee del mapeo objeto-relacional que relaciona las tablas de la base de datos con objetos de clases. Lo anterior se ve reflejado en la clase entidad del patrón DAO.

El patrón DAO nos apremia a crear uno por cada entidad de la base de datos que deseamos obtener información.

Las clases que contienen la tecnología de persistencia JPA son llamados repositorios, estas contienen las consultas a la base de datos. En el caso de los métodos que modifican la base de datos, estos son manejados gracias al soporte entregado por el módulo de Transacciones de Spring, para ello se establecen las anotaciones @Modifying y @Transactional. La anotación @Transactional es fundamental para llevar a cabo modificaciones en la base de datos sin poner en peligro su consistencia, ya que si el método anotado como transaccional falla, la transacción no se lleva a cabo, esto es de suma importancia cuando un método debe realizar varias modificaciones en la base de datos, debido a que la anotación permite cubrir todas estas modificaciones, lo que se conoce como “propagación”. En particular es usada la

característica de propagación de transacciones en la funcionalidad de creación de una entidad.

3.6.4. Vistas

Gracias a JavaScript, los contenidos de las vistas pueden ser dinámicos, de esta manera solo se ocupó una vista para mostrar la información necesaria y cubrir las funcionalidades del sistema. Además los eventos y las solicitudes al controlador, como también el despliegue de la información entregada por este último son manipulados por medio de JavaScript y AJAX. También nos permitió utilizar Leaflet para desplegar un mapa en la vista con la información geográfica.

La información resultante de las búsquedas se muestra en una tabla con la ayuda del plugin DataTables, que entre otras funcionalidades incluye la paginación, búsqueda dentro de la tabla y ordenar el contenido de las columnas.

Capítulo 4

Experimentos y Resultados

En esta sección se mostrarán las pruebas que se realizaron para medir el rendimiento del sistema, estas se llevaron a cabo en un computador con las siguientes especificaciones:

- Procesador: Intel(R) Core(TM) i5-6200U CPU 2.3Hz.
- RAM: 8 GB.
- Sistema operativo: Windows 10.
- Navegador Web: Google Chrome, version 73.0.3683.86 (Build oficial) (64 bits)

4.1. Tiempo de respuesta

Para medir el tiempo de respuesta del sistema se realizaron pruebas utilizando el tipo de búsqueda que contenga cualquiera de los nombres geográficos ingresados por el usuario en un campo de búsqueda total, determinado por el número de datos contenidos en la tabla geonames. Esta tabla al momento de realizar estas pruebas contenía 40.861 elementos.

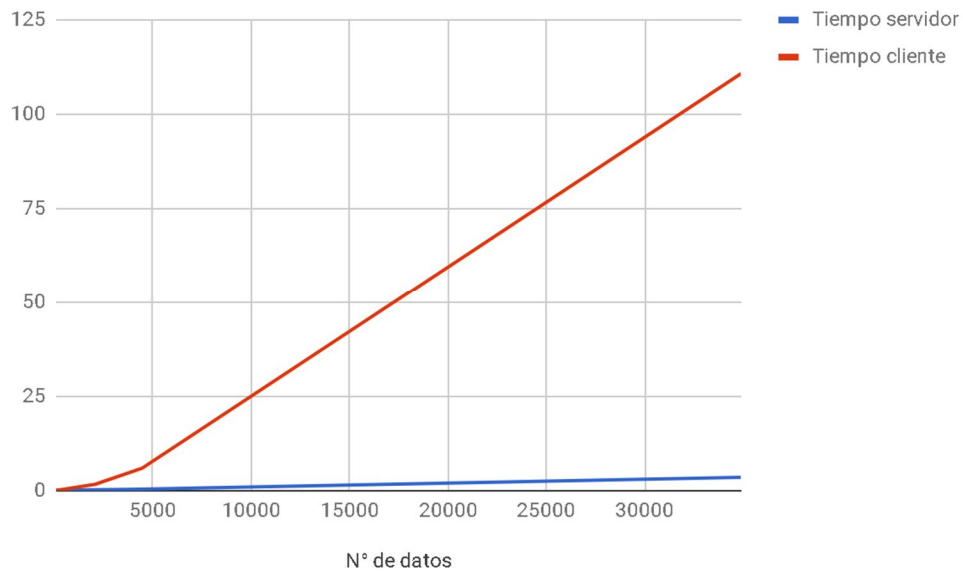
La elección de las características para realizar las pruebas de tiempo de respuesta se fundaron en considerar que este tipo de búsqueda es el más complejo a nivel de resultados que se pudiera obtener, ya que busca elementos de la tabla geoname que contengan lo buscado dentro del todo el espectro disponible.

Las pruebas se enmarcaron en medir el tiempo de respuesta de esta búsqueda del servidor y del lado del cliente.

Los resultados de las pruebas se pueden apreciar en la siguiente tabla:

	93 resultados		2079 resultados		4459 resultados	
N°	**Tiempo servidor (s)**	**Tiempo cliente (s)**	**Tiempo servidor (s)**	**Tiempo cliente (s)**	**Tiempo servidor (s)**	**Tiempo cliente (s)**
1	0,19	0,08	0,23	1,93	0,45	6,04
2	0,25	0,08	0,28	1,81	0,36	6,26
3	0,19	0,07	0,28	1,6	0,44	5,33
4	0,24	0,08	0,31	1,38	0,44	6,12
5	0,21	0,11	0,3	1,44	0,44	6,24
6	0,16	0,1	0,31	1,85	0,43	6,57
7	0,14	0,11	0,24	1,9	0,45	6,23
8	0,2	0,11	0,24	1,96	0,34	5,83
9	0,16	0,12	0,29	1,85	0,48	4,82
10	0,21	0,1	0,32	1,41	0,49	6,33
Promedio	**0,2**	**0,1**	**0,28**	**1,71**	**0,43**	**5,98**

De los datos obtenidos de las pruebas se puede observar que el tiempo de respuesta del servidor es relativamente bajo y está dentro de valores aceptables. En cambio las tiempos con respecto al cliente tienden a ser considerables en cuanto a los datos que debe interpretar y manejar de cara al usuario. Para lograr vislumbrar una tendencia de crecimiento se realizaron pruebas adicionales donde los datos resultantes de la búsqueda fueron de 34.875, al servidor le tomó en promedio 3,57 (s) en resolver la petición y al cliente le tomó en promedio 110,8 (s) para manipular los datos recibidos, lo cual resulta un tiempo por parte de esto último considerablemente alto y marca una tendencia exponencial. Todos los datos pueden ser apreciados en la siguiente gráfica.

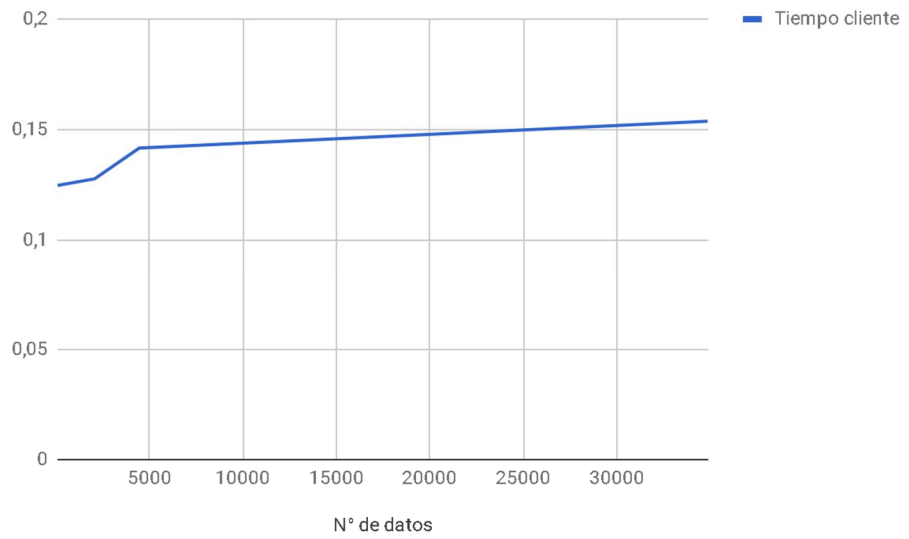


Gráfica 4.1: Tiempo v/s Número de datos de tiempos de espera cliente y servidor.

Teniendo en cuenta el mal desempeño por parte del cliente para manejar grandes volúmenes de datos se implementó la paginación provista por Spring del lado del servidor, lo que nos permite limitar los elementos entregados al cliente y la página que se desea mostrar independiente del número total de la búsqueda. Definiendo como máximo 100 elementos por página los resultados entregados por las pruebas de parte del cliente son los siguientes:

	93 resultados	2079 resultados	4459 resultados	34.875 resultados
N°	**Tiempo cliente (s)**	**Tiempo cliente (s)**	**Tiempo cliente (s)**	**Tiempo cliente (s)**
1	0,13	0,12	0,16	0,17
2	0,14	0,14	0,14	0,13
3	0,11	0,12	0,14	0,13
4	0,11	0,15	0,12	0,19
5	0,11	0,14	0,14	0,13
6	0,17	0,13	0,13	0,22
7	0,13	0,14	0,16	0,14
8	0,11	0,14	0,13	0,14
9	0,12	0,1	0,13	0,12
10	0,12	0,11	0,17	0,17
Promedio	**0,12**	**0,13**	**0,14**	**0,15**

Estos datos se reflejan en la siguiente gráfica:



Gráfica 4.1: Tiempo v/s Número de datos de tiempos de espera del cliente.

Gracias a la paginación el cliente se desempeña dentro de rangos aceptables. En cuanto al servidor no se observaron variaciones con respecto a lo medido anteriormente.

4.2. Pruebas unitarias

Spring nos permite, por medio de la herramienta JUnit, realizar de forma automática pruebas unitarias para comprobar que el comportamiento del sistema sea el deseado. Para ello se realizaron pruebas en diferentes secciones del sistema como en los controladores, algunas clases de la capa de persistencia y en algunas clases de de la capa de servicios. En el Apéndice C se incluye un esquema que ejemplifica las pruebas unitarias realizadas a nivel de repositorios.

Para llevar a cabo las pruebas se crean objetos o estructuras de prueba que simulan la información que se debe enviar o entregar en cada nivel.

En los controladores las pruebas se caracterizan por verificar que el nombre de la vista sea el correcto y para verificar que los datos esperados desde la capa de servicio sean realmente lo que debería entregar.

En la capa de servicio se prueban los métodos implementados por la clase encargada en el patrón DAO con el fin de verificar que estos están enviando y recibiendo la información de manera congruente según los casos de prueba utilizados. De la misma manera se realizaron pruebas en los repositorios.

Capítulo 5

Conclusiones

En esta Memoria de Título se desarrolló un sistema que implementa el servicio de Nomenclátor, el cual permite obtener información descriptiva a partir de nombres geográficos, por medio de distintas opcion que ayudan a acotar el campo de búsqueda para obtener la información que se desea.

Como objetivo general planteado se cumplió con el diseño e implementación del sistema que provee del servicio de Nomenclátor que permite el almacenamiento y gestión de información geográfica y descriptiva estandarizada de uso público y gratuito. Además el sistema fue implementado usando Software de código libre, donde se desarrolló una interfaz gráfica la cual permite llevar a cabo las funcionalidades que se establecieron y un servicio Web.

Parte de uno de los objetivos era que el sistema debe cumplir con el estándar WFS-G, el cual determina operaciones que entregan metadatos e información del servicio de Nomenclátor en formato XML. Este objetivo no se llegó a cumplir y se sugiere como trabajo futuro. Sin embargo, se desarrolló una versión de servicio Web que permite realizar peticiones HTTP obteniendo la información deseada en formato XML o JSON.

La interfaz gráfica del sistema permite realizar consultas definiendo opciones que especifican el tipo y campo de búsqueda. También posee las funcionalidades de agregar nueva información, modificar la ya existente y visualizar la información descriptiva y geográfica.

El sistema desarrollado en esta Memoria de Título busca ser una opción para solucionar la carencia de un servicio útil que provee de información importante que puede ser requerida para tratar o estudiar diversas problemáticas, como la seguridad, planificación y desarrollo, que aquejan a distintos niveles de la sociedad, tanto como a organismos públicos, privados y a la ciudadanía en general.

5.1. Trabajos Futuros

Durante el desarrollo del sistema se pensaron en diversas funcionalidades que pueden agregar mayor valor al sistema, por ejemplo:

- Gestión de usuarios con privilegios que les permitan o limiten el acceso a las distintas funcionalidades.

- Registro o historial más específico que pueda contener información con respecto a la creación y modificación.

- Más parámetros de búsqueda o creación.

- Exportar documentos en distintos formatos con la información solicitada al sistema.

- Implementar las operaciones necesarias para que el sistema cumpla con las especificaciones WFS-G.

Bibliografía

[1] Sitio Web: https://idearagon.aragon.es/buscadorNombresGeograficos/. Accedido por última vez el 13 de Marzo de 2019.

[2] Sitio Web: http://toponimia.xunta.es/Buscador. galicia Accedido por última vez el 13 de Marzo de 2019.

[3] Sitio Web: http://www.ideandalucia.es/nomenclator/buscador.jsp?lang=esp. Accedido por última vez el 13 de Marzo de 2019.

[4] López, M.; Luaces, M. R.; Paramá, J. R.: "Implementación de un Servicio de Nomenclátor según la norma MNE y el estándar WFS-G", en La Infraestructura de Datos Espaciales de España en 2007. Proyectos, servicios y nodos (JIDEE 2007), Grupo76, Santiago de Compostela (España), 2007, pp. 73-79.

[5] Miguel A. Bernabé-Poveda y Carlos M. López-Vázquez: “Fundamentos de las Infraestructuras de datos.”

[6] Sitio Web: http://download.geonames.org/export/dump/. Accedido por última vez el 26 de Marzo de 2019.

[7] Sitio Web: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/35. Accedido por última vez el 13 de Marzo de 2019.

[8] Sitio Web: https://www.tutorialspoint.com/spring. Accedido por última vez el 18 de Marzo de 2019.

[9] Sitio Web: https://www.baeldung.com/eclipse-sts-spring. Accedido por última vez el 20 de Marzo de 2019.

[10] Sitio Web: https://www.tutorialspoint.com/hibernate. Accedido por última vez el 20 de Marzo de 2019.

[11] Sitio Web: https://www.tutorialspoint.com/jpa. Accedido por última vez el 20 de Marzo de 2019.

[12] Sitio Web: https://www.tutorialspoint.com/maven. Accedido por última vez el 20 de Marzo de 2019.

[13] Sitio Web: https://www.tutorialspoint.com/junit. Accedido por última vez el 20 de Marzo de 2019.

[14] Sitio Web: https://leafletjs.com. Accedido por última vez el 11 de Marzo de 2019.

[15] Sitio Web: https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm. Accedido por última vez el 21 de Marzo de 2019.

[16] Sitio Web:
https://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm.
Accedido por última vez el 21 de Marzo de 2019.

[17] Sitio Web: http://www.un.org/spanish/esa/sustdev/agenda21/riodeclaration.htm. Accedido por última vez el 21 de Marzo de 2019.

[18] Sitio Web: http://tomcat.apache.org/. Accedido por última vez el 21 de Marzo de 2019.

[19] Sitio Web: https://www.postgresql.org/. Accedido por última vez el 28 de Marzo de 2019.

[20] Sitio Web: https://www.tutorialspoint.com/spring/spring_architecture.htm. Accedido por última vez el 28 de MArzo de 2019.

[21] Sitio Web: https://www.uv.es/grimo/teaching/SpringMVCv5PasoAPaso/part3.html. Accedido por última vez 29 de Marzo.

[22] Sitio Web: https://www.geonames.org/export/codes.html. Accedido por última vez 6 de Abril.

Apéndice A

Manual de Usuario

En siguiente manual se describen las funcionalidades del sistema tanto las que son realizadas en la interfaz gráfica como las consultas por URL.

A.1. Buscar Entidad:

La principal funcionalidad del sistema en la interfaz gráfica es la de buscar una entidad a partir de un nombre geográfico ingresado por el usuario. Los resultados obtenidos dependen del tipo de búsqueda, del nivel administrativo y si acota los resultados dentro de los límites del visor de mapa. La búsqueda por defecto se realiza en todo el territorio de Chile.

Están disponibles tres tipos de búsqueda según la coincidencia con el nombre geográfico ingresado, estas son: Coincidencia exacta, Comience por lo ingresado por el usuario y que contenga cualquiera de los nombres geográficos separados por coma. En la Figura A.1.1 se puede apreciar los campos para realizar la búsqueda.

Los niveles administrativos corresponden a las Regiones, Provincias y Comunas.

Figura A.1.1: Campos de Búsqueda, interfaz gráfica.

A.2. Resultados de la búsqueda

Los resultados de la búsqueda son desplegados en una tabla paginada en la cual permite realizar una búsqueda dentro de los resultados. Esto se puede apreciar en la Figura A.2.

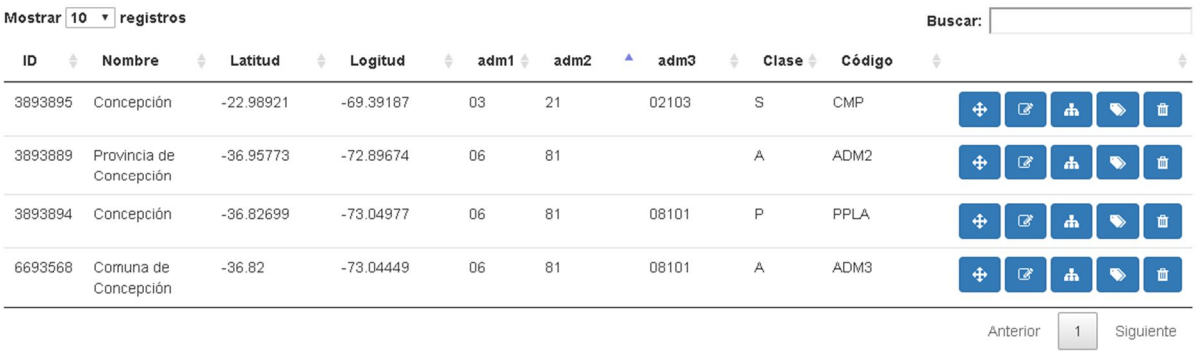


Figura A.2.1: Tabla de resultados de una búsqueda.

En la tabla de resultados cada elemento da acceso a la edición de la información de la entidad en la última columna izquierda. En esta columna nombrados de izquierda a derecha se puede realizar la edición de las coordenas, información general, niveles administrativos, edicion y creacion de nombres alternativos y eliminar la entidad.

A.3. Edición de una entidad

Cada elemento de la tabla de resultados de la búsqueda permite editar su información (ver Figura A.2.1), de izquierda a derecha, se le permite al usuario mover la entidad cambiando sus coordenadas, modificar información general, modificar sus niveles administrativos, editar nombres alternativos existente o crear nuevos y eliminar la entidad.

En la Figura A.3.1 se puede ver el panel de edición de coordenadas, una vez confirmada la acción, el sistema entregará información del estado de la petición, si se realizó con éxito, se verá modificado del marcador en el visor de mapa que corresponde al elemento editado a las nuevas coordenadas.

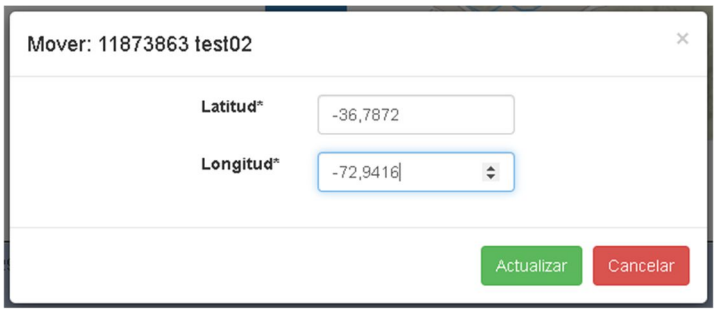


Figura A.3.1: Panel de edición de coordenadas.

La Figura A.3.2 muestra el panel de edición de información general, en él se pueden modificar el nombre de la entidad, la clase, el código de característica y la zona horaria.

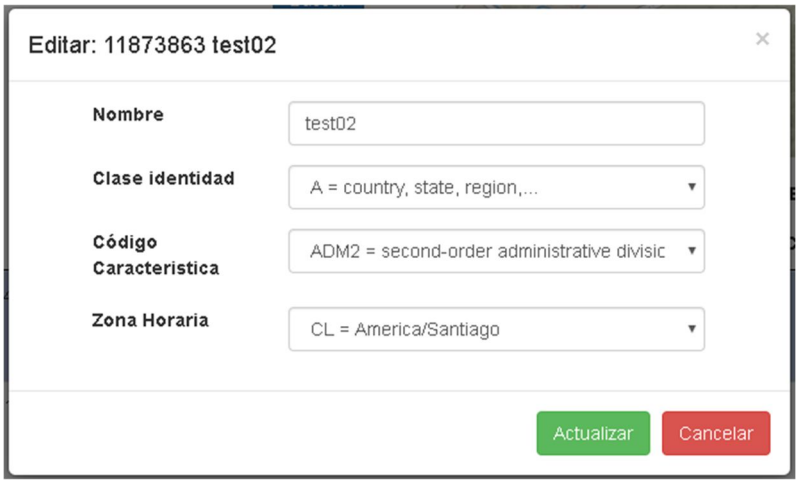


Figura A.3.2: Panel de edición de información general.

Al ingresar a la edición de los niveles administrativos, los campos muestran los actuales de la entidad y permitirá al usuario cambiarlos, existe una correspondencia de niveles que el sistema se encarga de llevar a cabo. En la Figura A.3.3 se puede apreciar este panel.

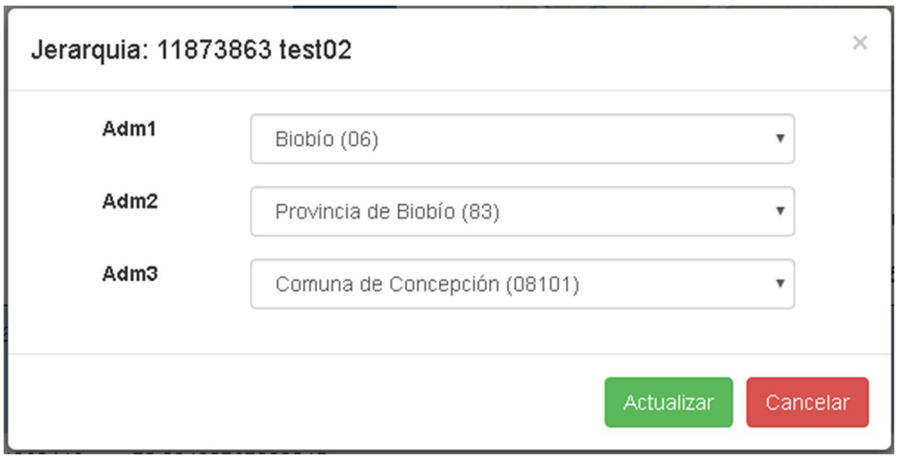


Figura A.3.3: Panel de edición de niveles administrativos.

La edición de nombres alternativos permite visualizar los ya existentes del elemento a editar (ver Figura A.3.4), además permite ingresar uno nuevo seleccionando el símbolo **+**, este desplegará, dentro del mismo panel, los campos a ser completados para ingresar otro nombre alternativo (ver figura A.3.5). Al confirmar la creación seleccionando el símbolo ✔, el sistema entregará información del estado final de la solicitud y se agregara la nueva información dentro del panel de nombres alternativos.

Figura A.3.4: Panel nombres alternativos.

Figura A.3.5: Campos para ingresar un nuevo nombre alternativo.

De manera similar descrito en la creación de un nombre alternativo se puede llevar a cabo la edición de los existentes seleccionando el primer símbolo de izquierda a derecha, este desplegará los mismos campos de la creación, pero con los valores actuales del elemento. Se puede eliminar un elemento seleccionando el icono de un basurero al lado del de edición, el sistema pedirá confirmación para esta acción y devolverá el panel de nombres alternativos actualizado.

Finalmente se puede eliminar toda la información relacionada a una entidad que se obtuvo como resultado de la búsqueda seleccionando el símbolo de un basurero de la última columna de la tabla de resultados. Este eliminará el elemento y su información previa confirmación del usuario. Al confirmar se volverá a mostrar los resultados de la búsqueda realizada.

A.4. Visor de mapa y creación de nueva entidad

Los resultados de la búsqueda se podrán apreciar con un marcado dentro del visor de mapa, pero también dentro del visor se implementó la funcionalidad de crear una una entidad, para acceder a ella se debe seleccionar con el botón derecho del ratón un punto del mapa, esto desplegará la opción de acceder a la funcionalidad (ver Figura A.4.1).

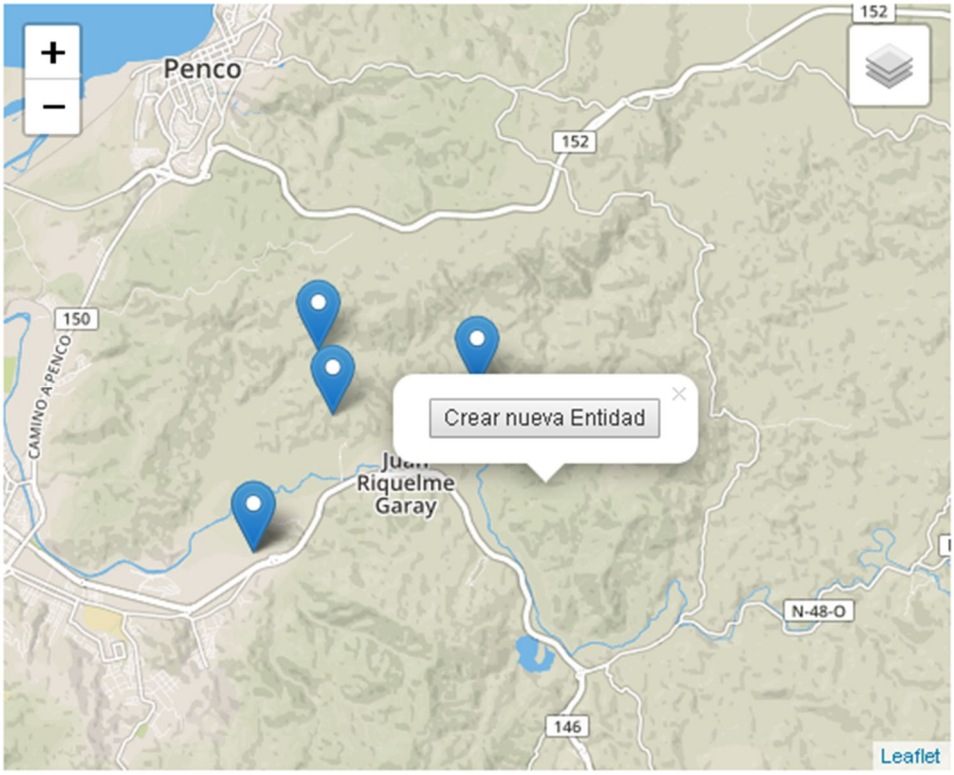


Figura A.4.1: Accediendo al panel de creación de entidad.

En la Figura A.4.2 se puede apreciar el panel de creación, en él se pueden agregar las coordenadas de la nueva entidad, que se establecen por defecto en el punto en que se accede al panel, también se puede agregar el nombre, los niveles administrativos, la clase, el código, los nombres alternativos ingresandolos separados por comas y la zona horaria.

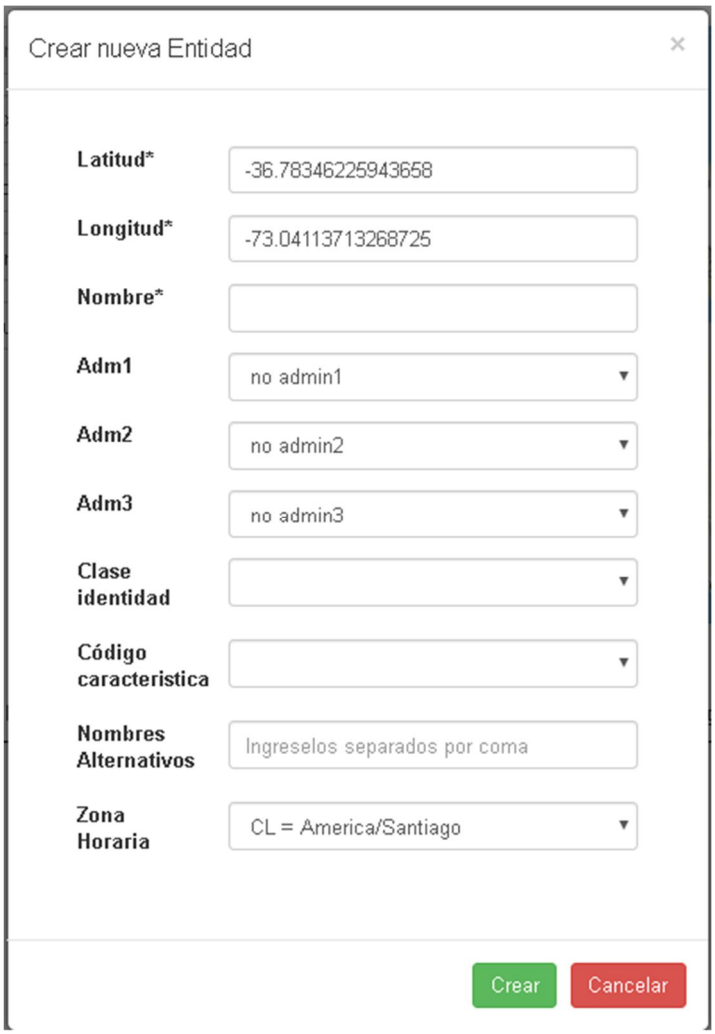


Figura A.4.2: Panel de creación de entidad.

A.5. Búsqueda XML o JSON

La búsqueda XML o JSON realiza una búsqueda similar a la realizada por medio de la interfaz gráfica, pero los resultados serán obtenidos en formato XML o JSON. Para acceder a estas funcionalidades se debe ingresar a los siguientes URLs (casos particulares):

http://localhost:8080/springapp/busquedaxml?
http://localhost:8080/springapp/busquedajson?

El primero devuelve los resultados en formato XML y el segundo en JSON. Para llevar a cabo la búsqueda se utilizan los siguientes parámetro en la URL:

- **nombre_igual:** Búsqueda de entidades con coincidencia exacta.
- **nombre_partePor:** Búsqueda de entidades que comiencen con el nombre geográfico.
- **nombre_contCual:** Búsqueda de entidades que contengan cualquiera de las palabras. Cada una de las palabras deben ser definidas de esta manera: nombre_contCual=nombregeografico&nombre_contCual=nombregeografico2&...
- **admin1:** Acota la búsqueda al primer nivel administrativo, Región. Se debe ingresar el código del nivel.
- **admin2:** Acota la búsqueda al segundo nivel administrativo, Provincia. Se debe ingresar el código del nivel.
- **admin3:** Acota la búsqueda al tercer nivel administrativo, Comuna. Se debe ingresar el código del nivel.
- **norte, sur, este, oeste:** Acota la búsqueda a un cuadro delimitado por estos parámetros. Deben ser usados los 4 a la vez.

Es obligatorio utilizar solo uno de los tres primeros parámetros anteriores; nombre_igual, nombre_partePor, nombre_contCual, los cuales se utilizan en conjunto con el nombre geográfico por el cual se realizará la búsqueda y su tipo. La búsqueda por defecto se realizará sin cotas de niveles administrativos, ni una zona delimitada geográficamente. Para acotar la búsqueda según niveles administrativos se debe ingresar uno de los parámetros; admin1, admin2, admin3. Si se desea acotar la búsqueda en una zona geográfica se deben ingresar todos los siguientes parámetros: norte, sur, este, oeste. Estas son las coordenadas de la zona limitante.

A continuación se muestran ejemplos para realizar estas búsquedas:

Ejemplo 1:
http://localhost:8080/springapp/busquedaxml?nombre_igual=Concepción&admin1=06

Ejemplo 2:
http://localhost:8080/springapp/busquedaxml?nombre_partePor=Cerro&admin3=06101&norte=-33.89&sur=-34.38&este=-70.31&oeste=-71.03

Apéndice B

Modelo Relacional

Geoname(geoname_id,nombre,nombre_ascii,latitud,longitud,clase_identidad,codigo_caracteristica,codigo_pais,codigos_pais_alternativos,codigo_admin1,codigo_admin2,codigo_admin3,poblacion,elevacion,modelo_elevacion_digital,zona_horaria_id,fecha_modificacion,ig)

pk: geoname_id

fk: zona_horaria_id de ZonaHoraria(zona_horaria_id)

clase_identidad de ClaseIdentidad(clase_identidad)

codigo_caracteristica de CodigoCaracteristica(codigo_caracteristica)

codigo_admin1 de ADMIN1(codigo_admin1)

codigo_admin2 de ADMIN2(codigo_admin2)

codigo_admin3 de ADMIN3(codigo_admin3)

Herencia(geoname_id_padre,geoname_id_hijo)

pk: geoname_id_padre, geoname_id_hijo

fk: geoname_id_padre de Geoname(geoname_id)

geoname_id_hijo de Geoname(geoname_id)

ZonaHoraria(zona_horaria_id,codigo_pais,GMT_offset,DST_offset,rawOffset)

pk: zona_horaria_id

ClaseIdentidad(clase_identidad,descripcion)

pk: clase_identidad

CodigoCaracteristica(codigo_caracteristica,clase_identidad,nombre,descripcion)

pk: codigo_caracteristica

fk: clase_identidad de ClaseIdentidad(ClaseIdentidad)

NombresAlternativos(nombre_alternativo_id,nombre_alternativo,isolenguaje)

pk: nombre_alternativo_id

Geoname_NombreAlternativo(geoname_id,nombre_alternativo_id)

pk: geoname_id, nombre_alternativo_id

fk: geoname_id de Geoname(geoname_id)

nombre_alternativo_id de NombresAlternativos(nombre_alternativo_id)

IsoLenguaje(iso_lenguaje_id,ISO_639-3,ISO_639-2,ISO_639-1,nombre_lenguaje)

pk: iso_lenguaje_id

ADMIN1(codigo_admin1,nombre,nombre_ascii,geoname_id)
pk: codigo_admin1

ADMIN2(codigo_admin2,nombre,nombre_ascii,geoname_id)
pk: codigo_admin2

ADMIN3(codigo_admin3,nombre,nombre_ascii,geoname_id)
pk: codigo_admin3

Apéndice C

Esquema prueba unitaria

A continuación se aprecia un esquema que ejemplifica las pruebas unitarias del sistema a nivel de repositorios.

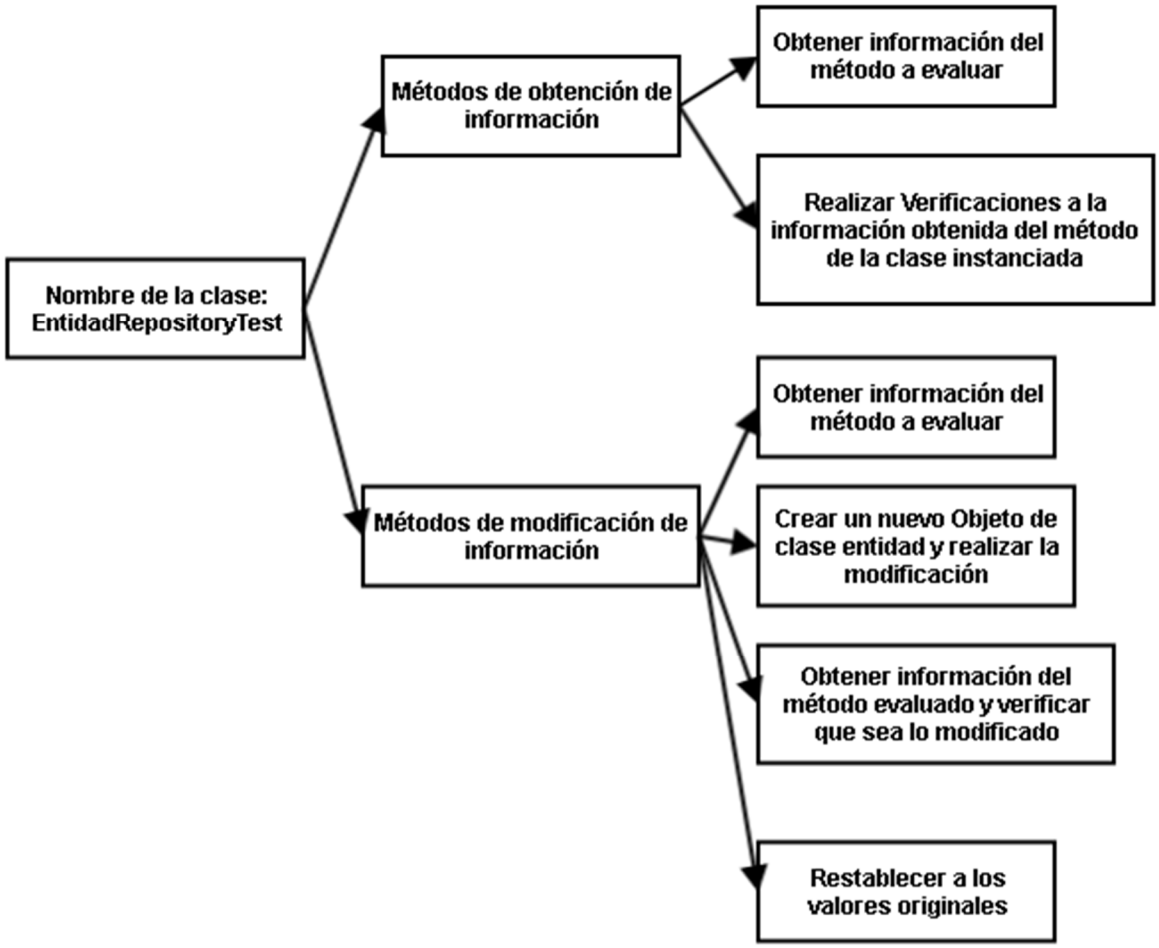


Figura C.1: Esquema de ejemplo de prueba unitaria nivel Repositorio.