Universidad de las Ciencias Informáticas

CITEC

API de servicios para el Módulo Configuración de SIGE

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Enmanuel Contreras Camejo

Tutora: MSc. Milenis Fernández Díaz

La Habana, noviembre de 2022

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Enmanuel Contreras Camejo, con carné de identidad 98020702529 soy el autor principal del trabajo de diploma titulado API de servicio para el Módulo de Configuración de SIGE y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firman la presente a los ___ días del mes de noviembre del año 2022.

_____

Firma del Autor

_____

Firma del Tutor

AGRADECIMIENTOS

Agradecer sobre todo a mi señor y protector, divino sobre todas las cosas. Único ser capaz de que supo levantarme en cada tropiezo y obstáculo hasta hoy. Gracias por todo.

A mi madre Mayelin, tan valioso tesoro que tengo, gracias por ser esa madre especial y dedicada te debo todo lo que soy y lo que llegaré a ser. Mamá en este día tan especial no es solo para mí, sino hoy cumplimos un sueño juntos, gracias por no soltarme la mano desde que di mis primeros pasos y por tenerla sujetada hasta el día de hoy, por el sacrificio, por las angustias que tantas veces pasamos, te debo esto y mucho más y no me alcanzan las líneas para decirte lo importante que eres para mí y si no lo sabes pues te lo digo, no me cansaré de decirte que te amo y que eres lo más importante en mi vida te quiero.

A mi novia Leydis, mejor amiga gracias por estar siempre a mi lado. Desde que nos conocimos siempre has estado ahí, para dar palabras de fuerza y ánimo en cada tropiezo que tuve en todo el trayecto de la universidad. Gracias mi vida, te amo.

Agradecer con todo mi corazón a mi abuelita Luisa, siempre fuisteis y siempre serás una mamá para mi. Abuelita, gracias por darme tanto amor y cariño en los años que convivimos juntos. Esos días fueron inolvidables para mí.

Quisiera expresar mi más sincero agradecimiento a todos los profesores que aportaron su grano de arena en la preparación de quien soy hoy en día.

A todos mis amigos que nunca me dieron la espalda y siempre tuvieron una sonrisa en sus rostros a la hora de contar con ellos y hablo de Jose Antonio, Josley, Manuel y Ariel, que bastante los mortifiqué en este tiempo.

En general a todos mis amigos, discúlpenme que no los mencioné uno a uno, si lo hiciera sería infinita la lista. A todos siempre los tendré en mi corazón.

DEDICATORIA

Este trabajo lo quiero dedicar a mi familia, en especial a mi madre por ser siempre mi apoyo y mi faro guía, a mi novia que siempre estuvo pendiente de todo, a mis abuelos que siempre me alentaron a que siguiera y no me cansara, a tirar hacia adelante, y a mis amigos que les toco la peor parte que fue soportarme todo este tiempo.

A todos muchas gracias.

RESUMEN

La investigación tuvo como objetivo desarrollar una API de servicio, que ofrece información asociada a los datos ya gestionados mediante el Módulo de Configuración de SIGE. La creación de esta aplicación viene dada por la importancia al sistema estadístico nacional la información referente al módulo de configuración, datos tales como: unidad de observación, registro, clasificadores, temáticas, aspectos, entre otros. Dicha información no esta disponible a las distintas Administraciones del Estado, por lo que no presentan un estudio estadístico uniforme, esto vuelve el trabajo estadístico por parte de estas instituciones más riguroso y tedioso, además de emplear recursos y gastos, en la confección de datos que pueden obtenerse de otros sistemas. Debido a esto se pretende con la API, llevar la información asociada del módulo a otros sectores de la institución, que sin importar la tecnología que estén utilizando, puedan obtener los recursos del módulo, para así lograr hacer análisis complejos en el país.

Para guiar el proceso de construcción de la propuesta de solución se utilizó la metodología de desarrollo de software Variación de AUP para la UCI y en la implementación se emplearon tecnologías y herramientas de software libre, tales como: PostgresSQL, como Sistema Gestor de Base de Datos; Django Rest como Framework de desarrollo de la aplicación; como IDE de desarrollo Visual Studio Code, Visual Paradigm for UML para el diseño de diagramas y además como herramienta de validación de la API de servicios la aplicación Postman, donde intervendrá en cada uno de los proceso de desarrollo del software.

Palabras claves: API, API REST, Módulo de Configuración, SIGE

ABSTRACT

The objective of the research was to develop a service API, which offers information associated with the data already managed through the SIGE Configuration Module. The creation of this application is given by the importance to the national statistical system of the information referring to the configuration module, data such as: observation unit, registry, classifiers, themes, aspects, among others. Said information is not available to the different State Administrations, so they do not present a uniform statistical study, this makes the statistical work by these institutions more rigorous and tedious, in addition to using resources and expenses, in the preparation of data that can be obtained from other systems. Due to this, the API is intended to take the information associated with the module to other sectors of the institution, which, regardless of the technology they are using, can obtain the resources of the module, in order to carry out complex analyzes in the country.

To guide the construction process of the solution proposal, the AUP Variation software development methodology for the UCI was used and free software technologies and tools were used in the implementation, such as: PostgresSQL, as the Database Management System ; Django Rest as Application Development Framework; as a Visual Studio Code development IDE, Visual Paradigm for UML for diagram design and also as a service API validation tool for the Postman application, where he will intervene in each of the software development processes.

Keywords: API, REST API, Configuration Module, SIGE

ÍNDICE DE CONTENIDO

INTRODUCCIÓN....13

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS DEL API DE SERVICIO WEB EN EL MÓDULO CONFIGURACIÓN DE SIGE....17

1.1 SISTEMA DE INFORMACIÓN ESTADÍSTICA EN CUBA....17

1.2 MÓDULO DE CONFIGURACIÓN DE SIGE....18

1.2.1 UNIDADES DE OBSERVACIÓN....19

1.2.2 CLASIFICADORES ESTADÍSTICOS....19

1.2.3 REGISTROS ESTADÍSTICOS....20

1.3 PROCESO DE INTEGRACIÓN DE SISTEMAS....20

1.3.1 API COMO FORMA DE INTEGRACIÓN DE SISTEMAS....21

1.3.3 VENTAJAS DE LAS API....21

1.3.4 TIPOS DE API....22

1.3.6 USO DE LA ESPECIFICACIÓN HTTP....23

1.3.7 PRINCIPAL VENTAJA DE UNA API REST....24

1.3.8 IMPLEMENTACIÓN DE SERVICIOS WEB CON REST....25

1.4 METODOLOGÍA Y HERRAMIENTAS EMPLEADAS EN LA SOLUCIÓN....25

1.4.1 Metodología de desarrollo....25

METODOLOGÍA VARIACIÓN DE AUP PARA LA UCI....26

1.4.2 LENGUAJE DE MODELADO....26

EL LENGUAJE UNIFICADO DE MODELADO....26

1.4.3 HERRAMIENTA CASE DE MODELADO....27

ÍNDICE DE CONTENIDO

VISUAL PARADIGM FOR UML....27

1.4.4 SISTEMA GESTOR DE BASE DE DATOS....27

POSTGRESQL....28

1.4.5 LENGUAJES DE PROGRAMACIÓN....28

PYTHON....28

1.4.6 MARCO DE TRABAJO....29

DJANGO REST FRAMEWORK....29

1.4.7 ENTORNO DE DESARROLLO INTEGRADO....29

VISUAL STUDIO CODE....29

1.4.8 HERRAMIENTA PARA VALIDACIÓN FUNCIONAL DE LA API 30

POSTMAN....30

CONCLUSIONES DEL CAPÍTULO....30

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL API DE SERVICIOS PARA EL MÓDULO DE CONFIGURACIÓN DEL SIGE....31

2.1 PROPUESTA DE SOLUCIÓN....31

2.1.1 DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN....31

2.1.2 MODELADO DE LA PROPUESTA DE SOLUCIÓN....32

2.2 LEVANTAMIENTO DE REQUISITOS....33

2.3 REQUISITOS FUNCIONALES....33

2.3 REQUISITOS NO FUNCIONALES....37

2.3 DESCRIPCIÓN DE REQUISITOS FUNCIONALES DE SOFTWARE....38

2.4 DISEÑO ARQUITECTÓNICO....40

2.4.1 ESTILO ARQUITECTÓNICO REST....40

2.5 PATRONES DE DISEÑO....41

2.5.1 PATRONES GRASP....41

2.5.2 PATRONES GOF....45

2.6 DIAGRAMA DE CLASES DEL DISEÑO....46

2.7 MODELADO DE DATOS....48

CONCLUSIONES....49

CAPÍTULO 3: IMPLEMENTACIÓN Y EVALUACIÓN DEL API DE SERVICIO PARA EL MÓDULO DE CONFIGURACIÓN DE SIGE....50

3.1 DIAGRAMA DE DESPLIEGUE....50

3.2 PRUEBAS DE SOFTWARE PARA LA EVALUACIÓN DE LA PROPUESTA DE SOLUCIÓN....50

3.3 APLICACIÓN DE LAS PRUEBAS DE SOFTWARE....51

3.3.1 PRUEBAS INTERNAS....51

PRUEBAS UNITARIAS....51

TÉCNICA DE PRUEBA: CAMINO BÁSICO....51

3.3.2 PRUEBAS FUNCIONALES....54

TÉCNICA DE PRUEBA: PARTICIÓN DE EQUIVALENCIA....54

DISEÑOS DE CASO DE PRUEBAS....54

CONCLUSIONES PARCIALES....58

CONCLUSIONES GENERALES....59

REFERENCIAS BIBLIOGRÁFICAS....60

ANEXOS....65

ANEXO 1: ENTREVISTA REALIZADA A ESPECIALISTAS DE SIGE....65

ANEXOS 2: HISTORIAS DE USUARIO....65

ANEXO 3 DISEÑO DE CASO DE PRUEBA.....70

ÍNDICE DE TABLAS

Tabla 1. Listado de los requisitos funcionales......33

Tabla 2. Listado de los requisitos no funcionales......37

Tabla 3 Historia de usuario del RF2 Mostrar listado de unidades de observación......38

Tabla 4. Historia de usuario del RF3 Búsqueda especifica de unidad de observación......39

Tabla 5. Diseño de casos de pruebas para el camino básico 1 del método listar.....53

Tabla 6. Caso de Prueba Funcional del RF2 Listar unidades de observación.....55

Tabla 7. Historia de usuario del RF1 Autenticar Usuario......65

Tabla 8. Historia de usuario del RF4 Mostrar listado de clasificadores. 66

Tabla 9. Historia de usuario del RF5 Buscar clasificadores estadísticos. 67

Tabla 10. Historia de usuario del RF6 Mostrar listado de registros estadísticos......68

Tabla 11. Historia de usuario del RF7 Buscar registros estadísticos......69

Tabla 12. Caso de Prueba Funcional del RF3 Buscar unidades de observación......70

Tabla 13. Caso de Prueba Funcional del RF4 Listar de clasificadores estadísticos......71

Tabla 14. Caso de Prueba Funcional del RF5 Buscar clasificadores estadísticos......72

Tabla 15. Caso de Prueba Funcional del RF6 Listar registros estadísticos......73

Tabla 16. Caso de Prueba Funcional del RF7 Buscar registros estadísticos.....74

ÍNDICE DE FIGURAS

Figura 1. Modelado de la propuesta de solución. 33

Figura 2. Estilo arquitectónico REST 41

Figura 3. Patrón controlador de la propuesta de solución. 42

Figura 4. Patrón de Alta cohesión de la propuesta de solución. 43

Figura 5. Patrón de Experto de la propuesta de solución. 44

Figura 6. Patrón de Bajo Acoplamiento de la propuesta de solución. 44

Figura 7. Patrón Decorador de la propuesta solución 45

Figura 8. Patrón Singlenton de la propuesta de solución. 46

Figura 9: Diagrama clase de diseño de listar unidades y búsqueda específica de unidades de observación. 47

Figura 10. Modelo de la base de datos. 48

Figura 11. Diagrama de despliegue de la propuesta de solución. 50

Figura 11. Código de la implementación de listar Clasificadores estadísticos. 52

Figura 12. Grafo de flujo del código de la implementación del método listar Clasificadores estadísticos. 52

Figura 13. Caso de Prueba Funcional de la historia de usuario Listar unidad de observación. Escenario 1.1 57

Figura 14. Caso de Prueba Funcional de la historia de usuario Listar unidad de observación. Escenario 1.2 58

Introducción

Con el triunfo de la Revolución en Cuba el 1ro de enero de 1959 comienza un profundo proceso de cambios y desarrollo de la economía y la sociedad. En consecuencia, a partir de la década del 60 del siglo pasado y hasta la actualidad, las estadísticas en Cuba han transitado por varias etapas, con el objetivo de registrar los hechos económicos, sociales y demográficos del país, donde su introducción ha sido clave en la evaluación y control de los planes de vida de la población.

Hoy por hoy la ONEI es el órgano fundamental en el proceso de desarrollo estadístico en el país, en la aplicación de censos, encuestas por muestreo y en el aprovechamiento de la información de los registros públicos y administrativos. Una de las herramientas utilizadas por esta organización es el Sistema Integrado para Gestión Estadística (SIGE), nacida por la necesidad de la ONEI de adquirir un sistema web que fuese gratuito.

SIGE se crea bajo la dirección de la Universidad de las Ciencias Informáticas (UCI) y del Centro de Tecnología de Gestión de Datos (DATEC). Es una herramienta desarrollada para mejorar la funcionalidad de los procesos estadísticos. Posee una arquitectura modular que permite la captura y recolección de datos, presentación de la información e inteligencia organizacional.

En cuanto a sus características, SIGE es desarrollado como una aplicación de software libre, promoviendo la integración y uso de varias tecnologías manejadas bajo este concepto. Entre ellas se encuentra: PostgreSQL, para la administración de bases de datos; Symfony; para optimizar la estructura y el desarrollo de esta aplicación en la web y la biblioteca de JavaScript Ext Js.

SIGE presenta 6 módulos principales, entre los cuales se encuentra el Módulo de Configuración. Este se encarga de gestionar el proceso de captación y procesamiento de la información estadística en relación con las unidades de observación. Dispone de registros que identifican a las distintas unidades de observación y de clasificadores que permiten agruparlas y caracterizarlas en un lenguaje adecuado (SIGE 2014), además permite gestionar datos como temáticas, indicadores, aspectos, sistema de información, entre otros, para un mejor estudio estadístico.

Dada la importancia de los datos estadísticos para el país, la ONEI como institución que coordina el manejo de las estadísticas en el contexto nacional, prevé la necesidad de dotar al SIGE de un mecanismo de integración. Dicho mecanismo facilitaría la comunicación con otras aplicaciones informáticas utilizadas por los organismos de la Administración Central del Estado, en la actualidad estas instituciones no tienen acceso a las clasificaciones y otros datos estadísticos realizados

mediante el SIGE, por lo que el manejo de la información estadística por parte de estas instituciones no es uniforme. Se quiere generar un sistema único de clasificaciones que permita contar con datos comparables en el tiempo, en el espacio y entre fuentes de información, pues esto resultaría fundamental para llevar a cabo análisis más complejos en el país.

Luego de analizar la situación problemática se identifica el siguiente **problema científico**:

¿Cómo exponer mediante servicios el uso de la información del Módulo de Configuración que permita su integración con otras aplicaciones informáticas?

Se plantea como **objeto de estudio:** la integración de sistemas, y como **campo de acción:** la integración de sistemas mediante la API de servicios web.

Para darle solución al problema anterior se plantea como **objetivo general** desarrollar la API de servicios que exponga la información del Módulo de Configuración de SIGE facilitando la integración con otras aplicaciones informáticas.

Se plantean los siguientes **objetivos específicos**:

1. Elaborar el marco teórico - referencial de la investigación relacionado con la integración de sistemas mediante API de servicios web.

2. Diseñar la API de servicios para Configuración de SIGE.

3. Implementar la API de servicios para Configuración de SIGE.

4. Validar las funciones de la API de servicios para Configuración de SIGE.

Para el cumplimiento de los objetivos antes expuestos, se definieron las siguientes **tareas de la investigación:**

1. Realización del estudio bibliográfico de las herramientas de diseño del Sistema de Configuración para identificar sus principales características.

2. Identificación de los principios y estilos arquitectónicos del Módulo de Configuración para el desarrollo de la API de servicios web.

3. Fundamentación de la selección de la metodología, las herramientas y las tecnologías a emplear en el desarrollo de la API de servicios web en módulo de Configuración.

4. Descripción de las clases conceptuales y asociaciones candidatas para el contexto de la API de servicios web.

5. Definición de los requisitos funcionales y no funcionales de la API de servicios web para el módulo de configuración para establecer lo que debe hacer y bajo qué circunstancias debe hacerlo.

6. Descripción del comportamiento detallado del sistema en función de los requisitos funcionales definidos para la API de servicios web en Modulo de Configuración.

7. Elaboración del modelo de diseño de la API de servicios web en Módulo de Configuración para detallar las clases software del diseño que definen el comportamiento del sistema.

8. Elaboración del modelo de implementación de la API de servicios web en Modulo de Configuración para describir los elementos del modelo de diseño y la estructuración de estos de acuerdo al entorno de desarrollo.

9. Realización de pruebas funcionales a la solución implementada.

Los **Métodos teóricos** utilizados para cumplir las tareas son los siguientes:

1. Histórico-Lógico: Se utiliza para el trabajo recopilatorio sobre el proceso de desarrollo de las API en la web.

2. Análisis y Síntesis: Se utilizó para identificar y analizar las diversas funcionalidades en el trabajo de las API en aplicaciones web que pueden ser aplicadas en la solución.

3. Modelación: Se utiliza para representar por medio de diagramas el proceso de desarrollo de la API propuesta, teniendo como resultado un mejor entendimiento de la posible solución a implementar.

Los **Método empírico** utilizado para cumplir las tareas fue:

Entrevista: individuales y colectivas: se emplea en encuentros con los trabajadores del SIGE para conocer la necesidad del desarrollo de la propuesta de solución, a su vez definir sus funcionalidades y restricciones que se imponen (Anexo 1).

Posibles resultados: API de servicio para brindar información del Módulo de Configuración de SIGE.

El documento está estructurado por tres capítulos:

Capítulo 1. Fundamentación teórica. En este capítulo se definieron conceptos importantes para el posterior entendimiento de la investigación. Se realizó una breve reseña y valoración sobre las características fundamentales de SIGE y del Módulo de Configuración, y se explicaron las herramientas, metodologías y lenguajes usados en la construcción de la solución.

Capítulo 2. Análisis y diseño de la API de servicios para el Módulo de Configuración de SIGE. Se definió la solución que se propone y se realizó la identificación de los requerimientos del sistema. Se analiza de la solución que se propone, se modelan y describen los recursos ingenieriles necesarios acorde a la metodología AUP-UCI que representan las funcionalidades del sistema, aplicando los patrones de arquitectura y diseño seleccionados.

Capítulo 3. Evaluación de la solución propuesta. En este capítulo se describe la implementación y posterior validación realizada al producto obtenido como solución.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS DEL API DE SERVICIO WEB EN EL MÓDULO CONFIGURACIÓN DE SIGE.

1.1 Sistema de Información Estadística en Cuba

En Cuba se evidencia esta parte de conocimiento con unos de sus principales actores en el proceso de captación y manejo de información estadística, la Oficina Nacional de Estadística e Información (ONEI). Esta institución tiene como principal objetivo: responder con calidad las necesidades de información estadística, para así enfrentar los requerimientos de los objetivos de desarrollo económico, social y medioambiental del país y su adecuado reflejo internacional.

Como objetivo central de esta Organización se trata de (ONEI):

• Perfeccionar el Sistema Nacional Estadístico.

• Implementar las modificaciones aprobadas para el Sistema de Información del Gobierno, SiGob.

• Orientar el desarrollo de las primeras del Sistema de Información del Gobierno, SiGob.

• Incrementar el nivel de efectividad del Sistema de Control Interno y el perfeccionamiento del Sistema de Auditoría.

Este proceso de estadística tiene gran relevancia en todo el país donde se han desarrollado distintos proyectos en esta disciplina tal es el caso de los diferentes censos de administración, como por ejemplo en:

• **Encuesta Nacional de Envejecimiento de la Población (ENEP- Enero de 2017)**

- Abarcó las 15 provincias del país, y el municipio especial Isla de la Juventud. Con una representatividad nacional, urbana y rural, y para cuatro regiones geográficas- de la población cubana de 50 años y más (cohortes de nacidos hasta 1966). Este estudio se inscribe en una línea priorizada de investigaciones del Centro de Estudios de Población y Desarrollo (CEPDE) de la Oficina Nacional de Estadística e Información (ONEI) en coautoría con el Centro de Investigaciones sobre Longevidad, Envejecimiento y Salud (CITED) del Ministerio de Salud Pública (MINSAP) (ONEI 2017).

• Encuesta Nacional sobre Igualdad de Género (ENIG-2016) Informe de Resultados Diciembre 2020.

- En noviembre de 2016, se realizó la Encuesta Nacional sobre Igualdad de Género (ENIG-2016), en las 15 provincias del país y el municipio especial Isla de la Juventud. La muestra seleccionada fue

diseñada de modo que también pudieran obtenerse resultados estadísticamente representativos por sexo (ONEI, 2016).

- Se considera que esta encuesta da continuidad a otros estudios sobre la igualdad de género en Cuba, llevados a cabo por diferentes instituciones.

- **Censo de Población y Viviendas 2012 Diciembre 2012**

- La Oficina Nacional de Estadística e Información (ONEI) pone a disposición de los usuarios la documentación metodológica y organizativa utilizada en el Censo de Población y Viviendas 2012 y las cifras preliminares resultantes de ese levantamiento (ONEI 2016).

Unas de las principales herramientas utilizadas en la realización de estos censos es el Sistema Integrado de Gestión Estadística (SIGE), este fue creada para servir de apoyo al procesamiento de información estadística en Cuba. Su aplicación se encuentra en cada uno de los municipios del país contribuyendo en gran mediada al desarrollo científico y técnico de la nación. SIGE contiene en su estructura a 6 módulos que conforman en gran parte al trabajo de este sistema. Uno de estos módulos es el de Configuración, a continuación se presenta algunas características fundamentales de este.

1.2 Módulo de Configuración de SIGE

El Módulo Gestión de Configuración tiene como principal objetivo desarrollar el proceso de captación y procesamiento de la información estadística en relación con las unidades de observación, para lo cual dispone de registros que identifican a las distintas unidades de observación y de clasificadores que permiten agruparlas y caracterizarlas en un lenguaje adecuado. Los clasificadores además ocupan un lugar en el orden conceptual y metodológico pues permiten identificar el alcance de los objetos clasificados. Además permite realizar su referencia geográfica de las unidades de observación gestionadas previamente en el mapa para realizar una búsqueda localizada y otras funciones necesarias (SIGE 2014).

Este proceso de gestión de información contiene tres componentes que no se pueden pasar por alto, y su conformación se centra en el módulo en sí, unidad de observación, clasificadores y registros estadísticos, por lo que su caracterización resulta de vital importancia en su conocimiento.

1.2.1 Unidades de observación

Unidad de observación se define como la unidad descrita por los datos que se analiza o sea refiere al qué o quién es objeto de interés en una investigación, la cual puede ser una persona, una vivienda, un establecimiento comercial, industrial o de servicio público, una explotación agrícola o ganadera, y su determinación depende del objeto de la investigación (Quevedo R. 2011).

La unidad debe ser clara, en tal forma que sea entendida por todos; además, adecuada al tipo de investigación; mensurable, que permita ser medida y comparable con los resultados obtenidos en investigaciones similares.

1.2.2 Clasificadores estadísticos

En el ámbito de la generación de información estadística resultan fundamentales, donde cada variable de observación debe contar con una clasificación, precisa y clara, que permita captar todos los valores posibles que puede tomar la variable. Este permite exponer la distribución del total de los elementos de un universo de estudio.

Las clasificaciones estadísticas se agrupan y organizan la información de manera significativa y sistemática, generalmente en conjuntos completos y estructurados de clases que son definidas por su similitud de acuerdo con un grupo de criterios. Además éstas son desarrolladas para apoyar y facilitar la recolección y organización de estadísticas.

Así, se puede sintetizar la importancia de las clasificaciones estadísticas en cuatro puntos (INEGI):

• Se utilizan para la generación, procesamiento y publicación de la información.

• Aseguran un manejo uniforme de los datos estadísticos.

• Permiten agregar y desagregar conjuntos de datos de una forma significativa para análisis complejos.

• Garantizan la comparabilidad con estadísticas producidas por uno o diversos proyectos en distintos tiempos de una sola institución o de otras instituciones nacionales o internacionales.

1.2.3 Registros estadísticos

Los registros estadísticos pretenden ser una lista completa de los objetos en un grupo de objetos específicos o de población. Sin embargo, los datos en algunos objetos pueden faltar debido a deficiencias en la calidad. Los datos sobre la identidad de un objeto deberían estar disponibles con el fin de actualizar y expandir el registro con nuevos valores de variables para cada objeto (FAO).

Una vez mencionadas principales características del Módulo de Configuración y los principales componentes que lo conforman resulta imprescindible abordar el proceso de integración los sistemas y por qué resultaría más ventajoso para una aplicación en desarrollo.

1.3 Proceso de Integración de Sistemas

Es un proceso que utiliza software para compartir información entre varios subsistemas de forma automática. Como cada sistema está programado con diferentes codificaciones, un integrador actúa como un intermediario que traduce los datos de cada software detrás de las escenas, siendo una interfaz entre sistemas (Henderson 2020).

Según la Asociación Española de Calidad, la integración de sistemas se define como el conjunto de elementos relacionados o que interactúan que permiten implantar y alcanzar la política y los objetivos de una organización, en lo que se refiere a aspectos diversos como pueden ser los de calidad, medio ambiente, seguridad y salud, u otras disciplinas de gestión.

Las ventajas más destacadas de este tipo de proceso son (EDSRobotics 2021):

- Eficacia y eficiencia en la gestión

Tanto a nivel equipo, como a nivel individual. La comunicación mejora y con ello el enfoque a resultados beneficiosos, tanto para el cliente final como para la empresa en general.

- Ahorro de recursos

No sólo económicos, sino de tiempo y humanos. Económicos en cuanto a desperdicio de dinero en fases, departamentos o áreas que estancan en todo el proceso de producción de un producto o servicio y, como consecuencia, lo encarecen. De tiempo en cuanto a malgasto de horas que pueden destinarse a formación o a reducción de jornadas laborales. Y finalmente, humanos en cuanto a satisfacción y sensación de realización a todos los trabajadores de una empresa.

- Disminución de la documentación

Si hay algo que ralentiza los procesos y desespera a los trabajadores de una organización es la burocracia. La integración de sistemas es el mayor enemigo de la burocracia y de las grandes cantidades de documentación que se originan como parte del proceso de creación de un producto o servicio. Por lo tanto, la integración de sistemas es la mejor aliada para aquellas organizaciones que persigan reducir al mínimo o, incluso, eliminar la documentación.

- Tareas y cometidos mejor distribuidos entre el personal

Una empresa correctamente organizada, integrada e interconectada es capaz de distribuir mejor las tareas de sus trabajadores, lo que repercutirá positivamente en su percepción en cuanto a su posición en la empresa.

- Decrecimiento en el surgimiento de errores

Desarrollar una integración de sistemas en una empresa no significa que no vayan a seguir cometiéndose errores. Sin embargo, esto proporcionará una mayor capacidad para solventarlos sin que tenga un coste económico importante.

1.3.1 API como forma de Integración de Sistemas

Una interfaz de programación de aplicaciones o API (del inglés *application programming interface*) representa una interfaz de comunicación entre componentes de software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general (UCA).

Por su fácil manejo y aplicación, las API resultan de lo más conveniente en su aplicación, por lo que la realización de este trabajo para la solución de la problemática planteada se llevará a cabo con esta forma de integración.

1.3.3 Ventajas de las API

Es importante referirnos cuando hablamos de esta forma de servicio a las diferentes ventajas en su aplicación entre las que se encuentran (UCA).

1. Hace más fácil el desarrollo de un programa, ya que debe proveer todos los conceptos para construirlo.
2. Está diseñado especialmente para los programadores, ya que garantiza que todos los programas que utilizan API, tendrán interfaces similares. Asimismo, esto le facilita al usuario aprender la lógica de nuevos programas.
3. Cuando se realiza una solicitud, el servidor llamará a la API, brindando la ventaja de disponer de una mayor cantidad de servicios.

1.3.4 Tipos de API

Las API se clasifican en dos grandes grupos: las API locales, que están pensadas para comunicar dos piezas de software dentro de un mismo entorno, y las API remotas, que están orientadas a crear un enlace de intercambio de información entre aplicaciones y servicios remotos a través de internet (Reyes 2020). Entre las API remotas encontramos las API como servicio web, estás ayudan al intercambio de información de un lado a otro, por medio de protocolos y estándares establecidos. Estos protocolos y estándares van a estar dados por REST, este se utiliza en el desarrollo de servicios bajo una arquitectura en red, mediante el protocolo HTTP. Dicho lo anterior para la solución de la problemática se emplea las API remotas, por estar orientadas a una comunicación por medio de internet, mediante un desarrollo de servicio web, con una diseño de arquitectura REST.

También podemos agrupar a las API por su política de disponibilidad: públicas y privadas. Las API públicas pueden ser empleadas por otros desarrolladores para potenciar sus proyectos mientras que las API privadas existen internamente para determinadas organizaciones (Reyes 2020). En el desarrollo de la aplicación se llevará acabo mediante un servicio privado, debido a la flexibilidad y importancia de los datos manejados mediante el módulo de configuración de SIGE, en el proceso de desarrollo estadístico en el país.

1.3.5 Arquitectura REST para el desarrollo de la API de servicio web

La API REST (por sus siglas en inglés de *Representational State Transfer*) es un tipo de arquitectura de desarrollo web que se apoya en el estándar HTTP, se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen como los recursos son definidos y diseccionados; se utiliza cuando se desea describir a una interfaz que transmite datos específicos de un domino sobre HTTP.

Los objetivos de este estilo de arquitecturan se centran en (Fernández, Alejandro y Santana 2016):

1. **Escalabilidad de la interacción con los componentes**. La web ha crecido grandemente sin degradar su rendimiento. Una prueba de ello es la variedad de clientes que pueden acceder a través de la web: estaciones de trabajo, sistemas industriales y dispositivos móviles.

2. **Generalidad de interfaces.** Gracias al protocolo HTTP, los clientes pueden interactuar con cualquier servidor HTTP sin ninguna configuración especial.

3. **Puesta en funcionamiento independiente.** Este hecho es una realidad que debe tratarse cuando se trabaja en Internet. Los clientes y servidores pueden ser puestos en funcionamiento durante años. Por tanto, los servidores antiguos deben ser capaces de entenderse con clientes actuales y

viceversa. Diseñar un protocolo que permita este tipo de características resulta muy complicado. HTTP permite la extensibilidad mediante el uso de las cabeceras, a través de las URLs, a través de la habilidad para crear nuevos métodos y tipos de contenido.

4. **Compatibilidad con componentes intermedios.** Los más populares intermediarios son varios tipos de proxy para la web y las cachés que se utilizan para mejorar el rendimiento. Otros permiten reforzar las políticas de seguridad donde destacan los *firewalls* y *gateway2* permitiendo encapsular sistemas no propiamente web. Por tanto, la compatibilidad con intermediarios permite reducir la latencia de interacción además de reforzar la seguridad y encapsular otros sistemas.

1.3.6 Uso de la especificación HTTP

En el desarrollo de una API REST es imprescindible conocer lo referente a la especificación HTTP, sobre todo a los métodos permitidos, códigos de estado y aceptación de tipos de contenido.

Los métodos son usados para manipular los diferentes recursos que conforman la API. Los principales métodos soportados por HTTP y por ello usados por una API REST son (Kurtz):

- POST: crea un nuevo recurso.

- PUT: modifica un recurso existente.

- GET: lista la información de un recurso.

- DELETE: elimina un determinado recurso.

- PATCH: modifica solamente un atributo de un recurso.

Estos métodos junto con la URI, proporcionan una interfaz uniforme que permite la transferencia de datos en el sistema REST aplicando operaciones concretas sobre un recurso determinado.

Cuando se realiza una petición determinada, es de vital importancia conocer si dicha operación se ha llevado a cabo de manera satisfactoria o por el contrario se ha producido algún tipo de error. Para ello, HTTP dispone de un amplio número de códigos de error/éxito que cubren todas las posibles respuestas que el usuario puede recibir cuando trata de manipular un recurso mediante el uso de una API REST.

Las más comunes son (Kurtz):

200 OK. Respuesta estándar para peticiones correctas.

201 Created. La petición haya sido satisfactoria en la creación de un nuevo recurso.

202 Accepted. La petición es correcta, pero este no ha sido completado.

400 Bad Request. La solicitud contiene sintaxis errónea.

403 Forbidden. La solicitud fue legal, pero el servidor rehúsa responder dado que el cliente no tiene los privilegios para hacer hacerla.

404 Not Found. Recurso no encontrado. Se utiliza cuando el servidor web no encuentra la página o recurso solicitado.

500 Internal Server Error. Es un código comúnmente emitido por aplicaciones empotradas en servidores web, cuando se encuentran con situaciones de error ajenas a la naturaleza del servidor web.

1.3.7 Principal ventaja de una API REST

La principal ventaja del uso de la API REST reside en la independencia que proporciona frente a cualquier consumidor, sin importar el lenguaje o plataforma con el que se acceda a ella. Esto permite que una misma API REST sea consumida por infinidad de clientes sea cual sea la naturaleza de estos y que el cambio a cualquier otro tipo de consumidor no provoque impacto alguno en ella (Alodesk 2020).

Esta característica proporciona fiabilidad, escalabilidad y una fácil portabilidad a cualquier otra plataforma, ya que aísla por completo al cliente del servidor. Sólo se requiere que el intercambio de información de las respuestas se haga en un formato soportado, por lo general JSON o XML. Dicha separación entre el cliente y el servidor hace que se pueda migrar a otros servidores o bases de datos de manera transparente, siempre y cuando los datos se sigan enviando de manera correcta (Alodesk 2020).

Esto convierte a las API REST en una de las arquitecturas web más utilizadas por la flexibilidad que aportan a cualquier entorno de trabajo, sea cual sea su naturaleza.

1.3.8 Implementación de servicios web con REST

REST hace que los desarrolladores usen los métodos HTTP explícitamente de manera que resulte consistente con la definición del protocolo. Este diseño básico establece una asociación uno-a-uno entre las operaciones de crear, leer, actualizar y borrar y los métodos HTTP. A continuación, se lista un conjunto de pasos lógicos necesarios su desarrollo (Fernández, Alejandro y Santana 2016):

1. Definir las URLs para cada funcionalidad que se desee realizar.

2. Se implementan las funcionalidades definiendo los parámetros de entrada y de salida. Estos servicios web generarán una respuesta a una petición realizada por un cliente del servicio, respuesta que generalmente será en formato JSON o XML.

3. Se publica el servicio web permitiendo visibilidad vía URL a las funcionalidades implementadas.

1.4 Metodología y herramientas empleadas en la solución

1.4.1 Metodología de desarrollo

La metodología de desarrollo constituye un conjunto de procedimientos, técnicas, herramientas y soporte documental que permite la creación de software. Sirven de guía en el proceso de desarrollo de software y se agrupan en dos grandes grupos, las tradicionales y las ágiles. Las tradicionales se enfocan en el plan del proyecto, estableciendo rigurosamente las actividades implicadas, los artefactos que se deben producir, además de las herramientas y notaciones que se utilizarán. Por otra parte las metodologías ágiles se centran en satisfacer las necesidades del cliente, dándole mayor valor al individuo.

Metodología Variación de AUP para la UCI

Existen numerosas metodologías entre las que están Programación Extrema o XP (*Extreme Programming*), RUP (*Rational Unified Process*), OpenUP entre otras. Para el desarrollo del presente trabajo fue utilizada la metodología Variación de AUP para la UCI, el cual es una variante de la metodología AUP *(Agile Unfied Process*, Proceso Unificado Ágil). La misma fue definida por la universidad para guiar el proceso de desarrollo de software en la institución ya que se adapta al ciclo de vida definido para la actividad productiva, se ajusta a las características de cada proyecto y lograr una mayor homogeneidad entre los procesos de desarrollo de todos los centros productivos de la UCI (Muñoz Gonzáles 2018).

Esta metodología contiene tres fases: Inicio, Ejecución y Cierre. El desarrollo de la propuesta de solución se centra en la fase de Ejecución ya que es donde se desarrollan las actividades para especificar los requisitos de software, diseñar, implementar y probar la solución.

En la elaboración de la herramienta informática propuesta, se emplea el escenario 4 de metodología, para la descripción de los requisitos ya que se aplica a los proyectos que tienen el negocio bien definido, el cliente acompaña al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos y el proyecto no es extenso.

1.4.2 Lenguaje de Modelado

El modelado de sistemas software es una técnica que ayuda al ingeniero de software a visualizar el sistema a construir. De ahí que los modelos pueden utilizarse para la comunicación con el cliente, grupo de desarrollo y otros factores que intervienen en el proceso de construcción de software, por lo que resulta imprescindible para cualquier proyecto (Erikson, Hans-Erik and Penker 1998).

El Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés de *Unified Modeling Language*) en su versión 2.0, permite especificar, construir, documentar y visualizar el modelado de clases, además de incluir los procesos de negocio y funciones del sistema. Define las API o componentes reutilizables, la construcción de los diagramas de clases, los componentes del sistema, despliegue e interacción.

Presenta capacidades para especificar comportamientos brindando apoyo a las metodologías de desarrollo de software abstrayéndose de los sistemas informáticos, soporta un diseño Orientado a Objetos (OO, por sus siglas en inglés de *Object Oriented*), donde se denotan en la representación visual del modelo aspectos como la visibilidad de los atributos o métodos, las relaciones entre clases y la variedad existente de las mismas (Grady, James y Ivar 2006).

Se decide utilizar UML debido a que es el lenguaje de modelado propuesto por la línea base de la arquitectura del Departamento Integración de Soluciones de CITEC, y además por las características antes mencionadas.

1.4.3 Herramienta CASE de Modelado

La tecnología CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadoras) supone la automatización del desarrollo del software, contribuyendo a mejorar la

calidad y la productividad en el desarrollo de sistemas de información. Permite automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas (Priscila, Gómez).

Visual Paradigm for UML

Es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm ha sido concebida para soportar el cicló de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Ofrece un lenguaje estándar común a todo el equipo de desarrollo, lo cual facilita considerablemente la comunicación. Permite la generación de código fuente y bases de datos mediante la transformación de diagramas de Entidad-Relación en tablas de base de datos. Posee licencia gratuita y se puede utilizar en múltiples plataformas (Pressman 2002).

A raíz de lo antes dicho resulta esencial el uso de esta herramienta, en su versión 8.0 para guiar el modelado de la solución.

1.4.4 Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD) o *DataBase Managenent System* (DBMS) es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de información del modo más eficiente posible (Marín 2019).

PostgreSQL

PostgreSQL es un sistema gestor de base de datos relacional OO, por lo que la información se representa mediante objetos. Este utiliza un modelo cliente/servidor y usa multiprocesos, lo cual garantiza la estabilidad del sistema. Su arquitectura brinda fiabilidad, integridad de datos y estabilidad (López 2016). Tiene una amplia comunidad de desarrollo en Cuba, principalmente en la UCI en el centro CREAD

Como gestor de base de datos se utilizará PostgreSQL en su versión 14.3, por las características antes mencionada y por ser el gestor de base de datos empleado por SIGE.

1.4.5 Lenguajes de Programación

Los lenguajes de programación son lenguajes artificiales que pueden utilizarse para definir una secuencia de instrucciones para su procesamiento por una computadora. Son herramientas que permiten crear programas y software. Los lenguajes de programación están formados por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura, además el significado de sus elementos y expresiones. Son utilizados para controlar el comportamiento lógico de una máquina y para expresar algoritmos con precisión. Estos lenguajes permiten ser leídos y escritos por personas y a su vez resultan independientes del modelo de computadora a utilizar (Juganaru 2014).

Python

Python es un lenguaje de programación potente y fácil de aprender. Tiene estructuras de datos de alto nivel eficientes y un simple pero efectivo sistema de programación orientado a objetos. La elegante sintaxis de Python y su tipado dinámico, junto a su naturaleza interpretada lo convierten en un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en muchas áreas, para la mayoría de plataformas.

El intérprete de Python y la extensa librería estándar se encuentran disponibles libremente en código fuente y de forma binaria para la mayoría de las plataformas desde la Web de Python y se pueden distribuir libremente. El mismo sitio también contiene distribuciones y referencias a muchos módulos libres de Python de terceros, programas, herramientas y documentación adicional.

El intérprete de Python es fácilmente extensible con funciones y tipos de datos implementados en C o C++ (u otros lenguajes que permitan ser llamados desde C). Python también es apropiado como un lenguaje para extender aplicaciones modificables (Python Software Foundation).

Como lenguaje de programación de escogió Python en su versión 3.10.3, por sus características antes mencionadas.

1.4.6 Marco de trabajo

Los marcos de trabajo en la informática son utilizados debido a que representan una estructura de software compuesta de componentes personalizables e intercambiables, lo que permite el desarrollo de aplicaciones más rápidas permitiendo optimizar el trabajo. Desde su creación han sido muy utilizados, debido a que facilita una mejor estructura y organización en los proyectos.

Django Rest Framework

Django Rest Framework es una aplicación Django que permite construir proyectos software bajo la arquitectura REST, incluye gran cantidad de código para reutilizar (*Views, Resources, etc*.) y una

interfaz administrativa desde la cual es posible realizar pruebas sobre las operaciones HTTP como lo son: POST y GET. Al ser una aplicación de Django, es basado en un marco web en Python, gratuito y de código abierto, de una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago (Docts Django).

Se utilizo Django Rest como framework en su versión 3.13.1 para el desarrollo del API de servicio por las características antes mencionadas.

1.4.7 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés de *Integrated Development Environment*), es un medio de desarrollo que ha sido empaquetado como un programa de aplicación, generalmente está compuesto por un conjunto de herramientas de programación en las que se encuentra el compilador, editor de código, depurador y constructor de interfaces gráficas. Proporciona un marco de trabajo amigable para diversos lenguajes de programación (Capterra).

Visual Studio Code

Visual Studio Code es un editor de código fuente. Tiene la característica de resaltado de sintaxis, finalización de código inteligente, fragmentos de código y refactorización de código entre otras, lo que ayudará grandemente a una mayor rapidez en el trabajo con el código. También es personalizable, de modo que los usuarios pueden cambiar el tema del editor, los métodos abreviados de teclado y las preferencias. Es gratuito y de código abierto.

Además el código combina la interfaz de usuario optimizada de un editor moderno con asistencia y navegación de código enriquecido y una experiencia de depuración integrada, sin la necesidad de un IDE completo (Damián de Luca 2015).

Para la implementación del sitio se utilizará este entorno de desarrollo en su versión 1.67.2 por las características antes mencionadas.

1.4.8 Pruebas de API de servicio

Las pruebas de API es una práctica que prueba el rendimiento, la confiabilidad, la seguridad y la funcionalidad de una API directamente a través de varias herramientas. Aquí pueden ver si una API puede cumplir con las expectativas realizando el mismo procedimiento que el usuario final para obtener la información.

A través de las pruebas de API, puede garantizar la posibilidad de obtener buenas respuestas en tiempo real a través de informes reales. Si una API devuelve la respuesta perfecta del formato esperado en el tiempo correcto, la calidad de la API es buena. El rendimiento y la calidad reales se pueden decidir realizando miles de llamadas simultáneamente (Prasad Acharya 2022).

Postman

Postman es una plataforma API para construir y usar API. Postman simplifica cada paso del ciclo de vida de la API y agiliza la colaboración para la creación de mejores API, más rápido (Postman).

Es una aplicación que nos permite realizar pruebas API. Es un cliente HTTP que nos da la posibilidad de testear 'HTTP requests' a través de una interfaz gráfica de usuario, por medio de la cual se obtiene diferentes tipos de respuesta que posteriormente deberán ser validados.

Para validación de la aplicación se utilizará Postman en su versión 10.11 por las características antes mencionadas.

Conclusiones del capítulo

En este capítulo se abarcó una serie de conceptos y características fundamentales que marcan el punto de partida para lograr un buen entendimiento de la resolución del problema. Se identificaron los principales beneficios que brindaría la aplicación de un proceso de comunicación entre varios sistemas. Así como forma de integración, los servicios API web, que desempeñan un papel primordial en su aplicación en la web, siendo capaz transmitir los datos de un lado a otro de forma cifrada, segura y ágil. Se definieron la principales características de la arquitectura REST, así como los beneficios en su utilización.

Se seleccionó AUP- UCI como metodología de desarrollo para guiar el ciclo de vida de la API del servicio. Además la herramienta Visual Paradigm 8.0 para el modelado del sistema, empleando el lenguaje de modelado UML 2.0. Se definió utilizar como IDE, el Visual Studio Code en su versión 1.63.2, utilizando como lenguaje de programación Python 3.10.3 soportados por los marcos de trabajos de Django- Rest- Framework en su versión 3.13.1.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE API DE SERVICIOS PARA EL MÓDULO DE CONFIGURACIÓN DEL SIGE.

2.1 Propuesta de solución

El módulo de configuración manipula y guarda información clave de gran interés en cuanto al procesamiento estadístico, entre los cuales se encuentra: unidades de observación, registros de las diferentes unidades, clasificadores, temáticas, aspectos, indicadores, sistema de información y pies de página. Dichos datos resulta esencial para el buen desarrollo tanto en el ámbito social y económico del país.

En el país muchas instituciones del Órgano Central no tienen acceso a los diferentes datos generados en cuanto a las clasificaciones estadísticas, por lo que el manejo de la información por parte de estas instituciones no es uniforme. Se quiere generar un sistema único de clasificaciones que permita contar con datos comparables en el tiempo, en el espacio y entre fuentes de información, pues esto resultaría fundamental para llevar a cabo análisis más complejos en el país.

Dicho lo anterior se propone como propuesta de solución el desarrollo de un mecanismo de integración; que facilitaría la comunicación con otras aplicaciones informáticas utilizadas por los Organismos de la Administración Central del Estado (OACE). La aplicación a realizar permite el acceso referente a los datos estadísticos generados por el Módulo de Configuración, que dependiendo de la solicitud, tener información disponible para su visualización posteriormente.

2.1.1 Descripción de la propuesta de solución

En el siguiente epígrafe se describen los objetos más importantes dentro del contexto del sistema a desarrollar, definiendo los conceptos más importantes que serán de utilidad para el posterior modelado de la propuesta de solución.

Unidad de observación: se refiere a qué o quién es objeto de interés en la investigación. Contiene datos relevantes como código, nombre descriptivo, fecha, detalles telefónico, email, dirección y registro al que está asociado.

Registro: permite identificar las unidades de observación. Contiene datos relevantes como código, alias y nombre descriptivo.

Clasificador: permite agrupar y organizar los registros en criterios de comparación definidos previamente. Contiene datos como código, nombre descriptivo, alias y dominio.

Indicador: permite establecer criterios de igualdad de las clasificaciones y las temáticas**.** Contiene datos como código, nombre descriptivo, unidad de medida, periodicidad, tipo de indicador, clasificación, temática.

Aspecto: se ocupa de reunir y organizar la información. Contiene datos como alias, nombre descriptivo y nota metodológica.

Sistema de información: se ocupa de reunir y gestionar información. Contiene datos como alias, denominación y logo.

Pie de firma: permite determinar la certificación y responsable de la información. Contiene datos como denominación, primer firmante, segundo firmante y nota de certificación.

Temática: indica diversos criterios de acuerdo a la investigación. Contiene datos como código y nombre descriptivo.

2.1.2 Modelado de la propuesta de solución

El siguiente diagrama muestra la relación entre los conceptos identificados en la propuesta de solución.

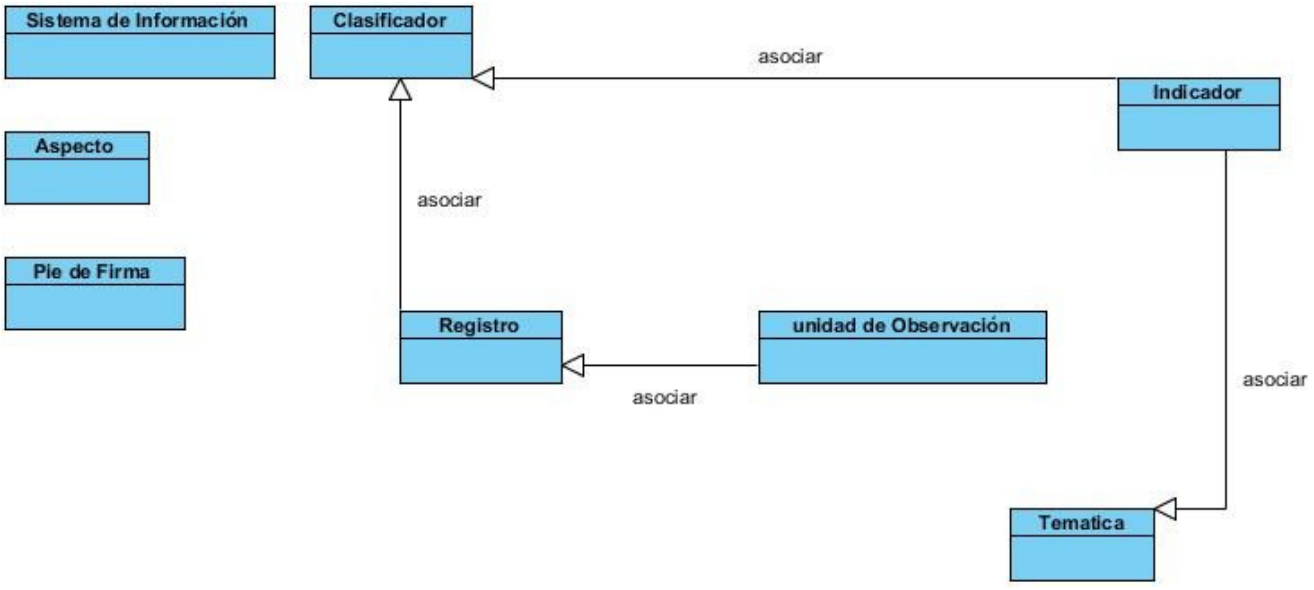


Figura 1. Modelado de la propuesta de solución.

2.2 Levantamiento de requisitos

Un requisito, o también denominado requerimiento, es definido por el glosario de la IEEE como una condición o capacidad que debe satisfacer o poseer un sistema o un componente de un sistema para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto. La definición y descripción de estos, ayuda a que exista una mayor comunicación entre el cliente y el equipo de desarrolladores. Existen dos tipos de requisitos: los requisitos funcionales y los no funcionales.

2.3 Requisitos funcionales

Los requisitos funcionales definen el comportamiento interno de un software, define lo que este debe hacer a un determinado evento. Para el desarrollo de la solución propuesta, el sistema debe cumplir con los siguientes requisitos funcionales. Ver tabla 1. Listado de requisitos funcionales.

Tabla 1. Listado de los requisitos funcionales.

Número	**Requisito funcional**	**Descripción**	**Complejidad**
RF 1.	Autenticar usuario	El sistema debe ser capaz de validar credenciales, para acceder a los servicios de la aplicación	Alta
RF 2.	Listar unidades de observación	El sistema debe permitir mostrar un listado de las unidades de observación.	Alta
RF 3.	Buscar unidades de observación	El sistema debe ser capaz de hacer una búsqueda específica de las unidades de observación listadas.	Alta
RF 4.	Listar clasificadores estadísticos	El sistema debe permitir mostrar el listado de clasificadores estadísticos.	Alta

RF 5.	Buscar clasificadores estadísticos	El sistema de ser capaz de hacer una búsqueda específica de clasificadores.	Alta
RF 6.	Listar registros	El sistema debe permitir mostrar una lista de registros, identificado en cada una de ellas sus diferentes atributos.	Alta
RF 7.	Buscar registros estadísticos	El sistema debe ser capaz de hacer una búsqueda específica de registros, mostrando el registro dado, con sus diferentes datos.	Alta
RF 8.	Listar indicadores	El sistema debe permitir mostrar una lista de indicadores.	Alta
RF 9.	Buscar indicadores	El sistema de ser capaz de hacer una búsqueda específica de indicadores.	Alta
RF 10.	Listar temáticas	El sistema debe permitir mostrar una lista de temáticas.	Alta
RF 11.	Buscar temáticas	El sistema de ser capaz de hacer una búsqueda específica de temáticas.	Alta
RF 12.	Listar aspectos	El sistema debe permitir mostrar	Alta

		una lista de aspectos.	
RF 13.	Buscar aspectos	El sistema debe ser capaz de hacer una búsqueda específica de aspectos.	Alta
RF 14.	Listar pies de firma	El sistema debe permitir mostrar una lista de pies de firma.	Alta
RF 15.	Buscar pies de firma	El sistema debe ser capaz de hacer una búsqueda específica de pies de firma.	Alta
RF 16.	Listar sistema de información	El sistema debe permitir mostrar una lista detallada de los diferentes sistemas de información.	Alta
RF 17.	Buscar sistema de información	El sistema debe ser capaz de hacer una búsqueda específica de una unidad sobre un conjunto de sistemas de información.	Alta

2.3 Requisitos no funcionales

Los requisitos no funcionales son requerimientos de calidad, que representan restricciones o las cualidades que el sistema debe tener tales como: precisión, usabilidad, seguridad, rendimiento,

confiabilidad entre otras (Melina Hernández, Granda Dihigo y Velázquez Cintra 2019). En cuanto a estas características, la aplicación a desarrollar se presentará con los siguientes requisitos:

Tabla 2. Listado de los requisitos no funcionales.

Número	**Requisitos no funcionales**	**Descripción**
RNF 1.	Usabilidad	Podrá estar accesible a cualquier sistema informático, que pueda hacer peticiones a la aplicación mediante el protocolo HTTP.
RNF 2.	Disponibilidad	Los servicios de la aplicación deben estar disponible en todo momento.
RNF 3.	Confidencialidad	La información manejada por el sistema debe estar protegida ante el acceso no autorizado.

2.3 Descripción de requisitos funcionales de software

La descripción de los requisitos funcionales de la propuesta de solución se realizará mediante las historias de usuario. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla (Jeffries, Anderson y Hendrickson 2001). A continuación, se muestra la historia de usuario del RF2 y RF3 que preside de una prioridad alta, las restantes historias de usuario se mostrarán en los anexos.

Historia de Usuario para RF: 2

Tabla 3 Historia de usuario del RF2 Mostrar listado de unidades de observación.

Historia de Usuario

Número: 1	**Nombre del requisito:** Listar unidades de observación.
Programador: Emanuel Contrera	**Iteración Asignada:** 1 iteración
Prioridad: Alta	**Tiempo Estimado:** 2 semanas
Riesgo de Desarrollo: No aplica	**Tiempo Real:** 2 semanas
Descripción: La aplicación muestra un listado de unidades de observación con sus diferentes atributos, listados de forma ascendente por su número de código. Las unidades de observación presentarán atributos como: código, nombre descriptivo, registro, estado, provincia y municipio.	
Observaciones: ● El usuario debe autenticarse en el sistema. ● Para acceder a la URL debe utilizar el método GET. ● La URL para acceder a la funcionalidad está compuesta por: La dirección donde esté publicado el servicio. /listar/unidades/	

Historia de Usuario para RF: 3

Tabla 4. Historia de usuario del RF3 Buscar unidad de observación.

Historia de Usuario	
Número: 2	**Nombre del requisito:** Buscar unidades de observación
Programador: Emanuel Contrera	**Iteración Asignada:** 1 iteración
Prioridad: Alta	**Tiempo Estimado:** 2 semanas

Riesgo de Desarrollo: No aplica	**Tiempo Real:** 2 semanas
Descripción: La aplicación muestra la unidad de observación o unidades, del conjunto del listado de unidades de acuerdo al criterio de búsqueda. El criterio de búsqueda estará especificada por el nombre descriptivo de la unidad de observación seleccionada. Una vez encontrada la especificación de la búsqueda; la aplicación mostrará la unidad de observación o unidades con sus diferentes atributos: código, nombre descriptivo, registro, estado, provincia y municipio.	
Observaciones: ● El usuario debe autenticarse en el autenticarse. ● Para acceder a la URL debe utilizar el método GET. ● La URL para acceder a la funcionalidad está compuesta por: La URL donde esté publicado el servicio Ejemplo: /listar/unidades/?search=sistema de finanzas/	

2.4 Diseño arquitectónico

El diseño arquitectónico permite definir cómo debe organizarse un sistema y cómo tiene que diseñarse la estructura global de este. Es el enlace entre el diseño y la ingeniería de requerimientos, que identifica los principales componentes estructurales en un sistema y la relación entre ellos (Garlan y Shaw 1993). En la propuesta de solución se utilizó un estilo de arquitectura REST.

2.4.1 Estilo arquitectónico REST

El estilo arquitectónico REST permite que las comunicaciones entre productor y consumidor sean más ligeras, mantenibles y escalables. Se emplea para el desarrollo de API REST empleando el protocolo HTTP, garantizando que pueda ser utilizada prácticamente por cualquier lenguaje de programación y de manera fácil. Además, es un requisito de un servicio REST que el cliente y el servidor sean independientes entre sí. En la siguiente figura se muestra un ejemplo de la arquitectura REST.

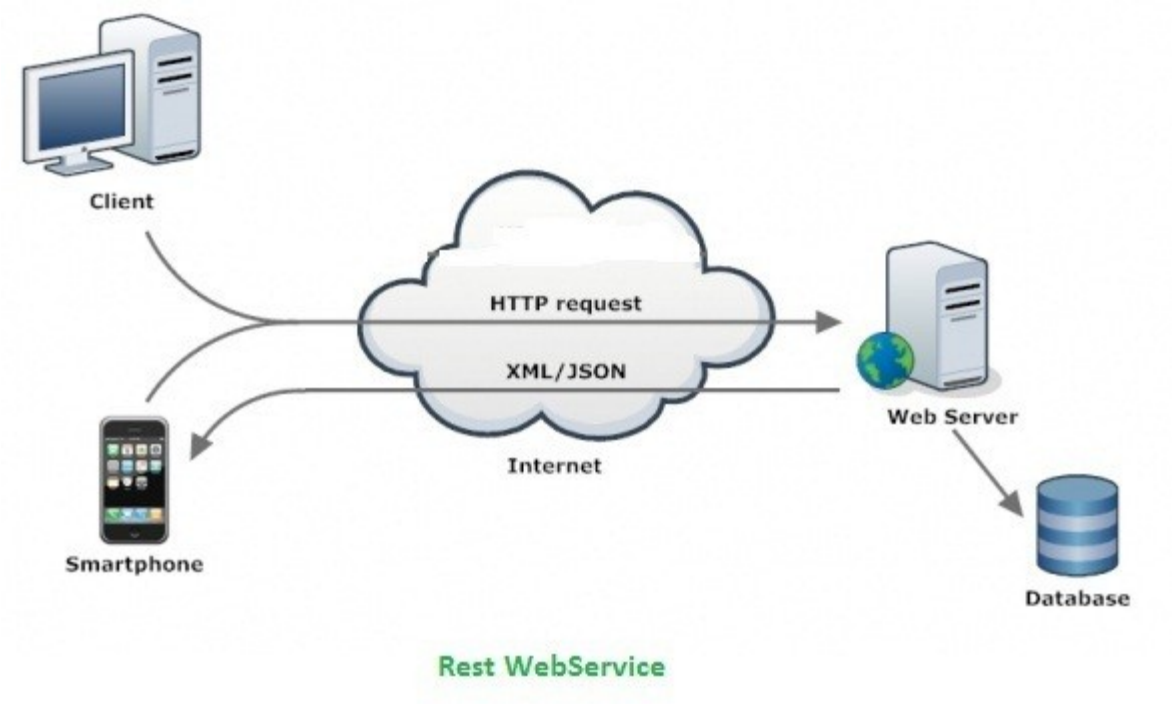


Figura 2. Estilo arquitectónico REST. (Fuente: Parvez. Types of Web Services SOAP,XML-RPC and Restful. [En línea] 2017. https://www.phpflow.com/php/web-service-types-soapxml-rpcrestful)

2.5 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptado para resolver un problema de diseño general en un contexto particular. Constituyen parte de la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Entre los patrones de diseño encontramos los patrones denominados Banda de los 4 (GOF, por sus siglas en inglés de *Gang Of Four*) y los Patrones de Asignación de Responsabilidades (GRASP, por sus siglas en inglés de *General Responsibility Assignment Software Patterns*) (Craig 1999).

2.5.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de

objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. El uso de estos patrones es de vital importancia para el desarrollo de una aplicación con la calidad requerida (Craig 1999). A continuación, se muestran los patrones GRASP utilizados en la propuesta de solución:

Controlador: Es un evento generado por actores externos. Se asocian con operaciones del sistema, operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos [31]. En la siguiente figura se puede observar el uso de este patrón, en la clase clasificadoresViewSet pues esta es la encargada de dar respuesta a una determinada acción por parte del cliente, en este caso listar o buscar los clasificadores estadísticos.

```
from rest_framework.routers import DefaultRouter
from apps.modulo_configuracion.api.api import *
router= DefaultRouter()
router.register(r'clasificadores', clasificadoresViewSet, basename='Clasificadores')

class clasificadoresViewSet(generalListRetrieve):
    serializer_class = clasificadores_serializer
    filter_backends = [filters.SearchFilter]
    search_fields = ['nombre_descriptivo', 'alias']
```

Figura 3. Patrón controlador de la propuesta de solución.

Alta cohesión: Asigna responsabilidades de manera tal que la cohesión siga siendo alta, o sea que las funcionalidades de las clases estén altamente relacionadas de forma tal que exista una colaboración entre ellas para compartir el esfuerzo y no caiga todo el peso sobre una única clase. Usar este patrón simplifica el mantenimiento y favorece el bajo acoplamiento (Craig 1999). Ejemplo de ello se puede ver en la siguiente figura 4 donde la clase clasificadoresViewSet hereda de la clase generalList, que contiene el método list, el cual se encarga de listar los clasificadores estadísticos.

```
class generalList(Authentication, viewsets.ReadOnlyModelViewSet):
    serializer_class = None

    def get_queryset(self):
        return self.get_serializer().Meta.model.objects.all()

    def list(self, request, *args, **kwargs):
        queryset = self.filter_queryset(self.get_queryset())
        serializer = self.get_serializer(queryset, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)

class clasificadoresViewSet(generalList):
    serializer_class = clasificadores_serializer
```

Figura 4. Patrón de Alta cohesión de la propuesta de solución.

Experto: Es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. Ofrece una analogía con el mundo real (Craig 1999). Ejemplo de este patrón se evidencia en la clase unidades_observacion, es la que clase que se responsabiliza del acceso a la información que le corresponde.

```
class unidades_de_observacion(models.Model):
    codigo=models.AutoField(primary_key=True)
    nombre_descriptivo=models.CharField(max_length=220, unique=True)
    Registro=models.ForeignKey(registro, on_delete=models.CASCADE, verbose_name= 'Registro')
    fecha=models.DateField()
    detalles_Teléfono=models.IntegerField(unique=True)
    email=models.EmailField('Correo Elecronico', max_length=225, unique=True)
    dirección= models.CharField(max_length=220)

    class Meta:
        verbose_name='Unidad de Observación'
        verbose_name_plural='Unidades de Observación'

    def __str__(self):
        return self.nombre_descriptivo
```

Figura 5. Patrón de Experto de la propuesta de solución.

Bajo acoplamiento: El Bajo Acoplamiento es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto

acoplamiento (Sixto 2003). Ejemplo del uso de este patrón se evidencia en la clase generalList, ya que se relaciona con la menor cantidad de clases posibles para el cumplimiento de su función.

```
class generalList(Authentication, viewsets.ReadOnlyModelViewSet):
    serializer_class = None

    def get_queryset(self):
        return self.get_serializer().Meta.model.objects.all()

    def list(self, request, *args, **kwargs):
        queryset = self.filter_queryset(self.get_queryset())
        serializer = self.get_serializer(queryset, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)
```

Figura 6. Patrón de Bajo Acoplamiento de la propuesta de solución.

2.5.2 Patrones GOF

Los patrones GoF se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro “Design Patterns—Elements of Reusable Software” de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides, a partir de entonces estos patrones son conocidos como los patrones de la pandilla de los cuatro (Fabbri). Se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

Decorador: Asigna responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la herencia para extender la funcionalidad (Sixto 2003). En la Figura 7 se presenta el uso de este patrón en la propuesta de solución. Ejemplo del uso de este patrón se evidencia en la clase UserToken.

```
from rest_framework.authtoken.models import Token

class Usertoken(APIView):
    def get(self, request, *args, **kwargs):
        username= request.GET.get('username')


        try:
            user_token= Token.objects.get(
                user= UserSerializer().Meta.model.objects.filter(username= username).first()
            )

            return Response({
                'token': user_token.key
            })
        except:
            return Response({
                'error': 'Credenciales enviadas incorrectas.'
            }, status= status.HTTP_400_BAD_REQUEST)
```

Figura 7. Patrón Decorador de la propuesta solución

Singleton: Es un patrón de diseño que se utiliza para garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella (Fabbri). En la aplicación es imprescindible que exista solamente una instancia de los ficheros que se van a utilizar para leer la información almacenada en el repositorio. Esta información va a ser accedida desde un solo punto, permitiendo un acceso controlado a la única instancia. Se va a generar un archivo donde se almacenan los mismos, de este archivo es necesario tener una única instancia que se va a utilizar para almacenar en la base de datos la información contenida en él. Ejemplo de este tipo de diseño es en la clase unidad_observacion.

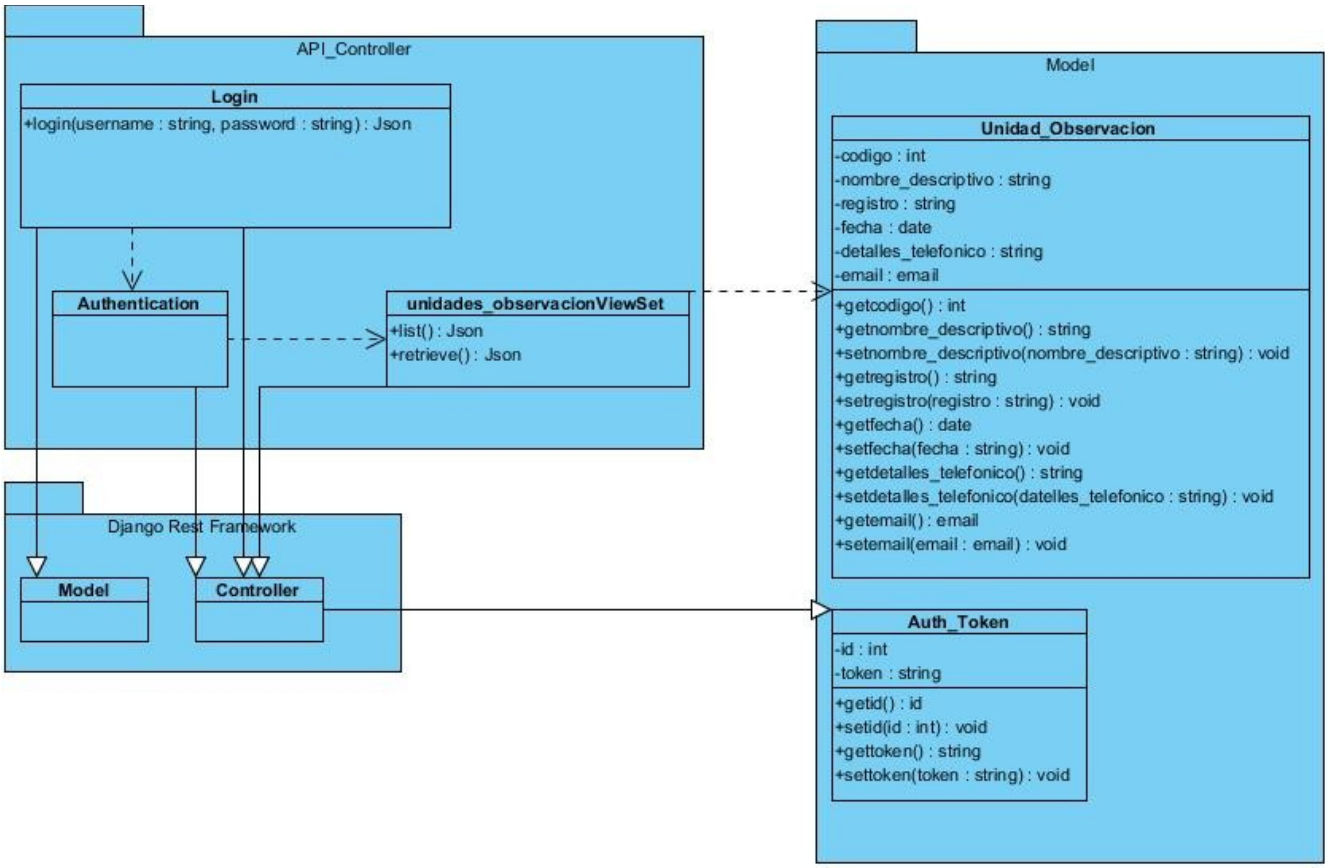
```
class unidades_de_observacion(models.Model):
    codigo=models.AutoField(primary_key=True)
    nombre_descriptivo=models.CharField(max_length=220, unique=True)
    Registro=models.ForeignKey(registro, on_delete=models.CASCADE, verbose_name= 'Registro')
    fecha=models.DateField()
    detalles_Teléfono=models.IntegerField(unique=True)
    email=models.EmailField('Correo Elecronico', max_length=225, unique=True)
    dirección= models.CharField(max_length=220)

    class Meta:
        verbose_name='Unidad de Observación'
        verbose_name_plural='Unidades de Observación'

    def __str__(self):
        return self.nombre_descriptivo
```

Figura 8. Patrón Singlenton de la propuesta de solución.

2.6 Diagrama de clases del Diseño

El Diagrama de Clases del Diseño es uno de los artefactos que genera el Modelo de Diseño, en él se muestran un conjunto de clases, interfaces y colaboraciones, así como las relaciones entre sí. Se realiza un diagrama por caso de uso identificado. Estos diagramas son importantes pues visualizan, especifican y documentan los modelos estructurales, lográndose una muestra amplia y confiable del sistema previo a su implementación.

Diagrama de Clases del Diseño: Listar Unidades de Observación y búsqueda de Unidad de Observación.



En la figura muestra cómo se inicia en el proceso del consumo de la API, con una primera instancia con una validación de autenticación de usuario. La clase controladora Authenticacion va a heredar de la clase controladora propia de Django- Rest- Framework y en ella se encuentran implementadas varias herramientas para brindar los servicios siguiendo el estándar REST. Seguido el proceso de validación de credenciales, se hace uso de los métodos permitidos por la API de servicios, tanto el listado de unidades de observación, como una búsqueda específica de unidades de observación.

2.7 Modelado de datos

El modelado de datos es el proceso de crear un diagrama simplificado de un sistema de software y los elementos de datos que contiene, utilizando texto y símbolos para representar los datos y cómo fluyen. Los modelos de datos proporcionan un modelo para diseñar una nueva base de datos o rediseñar una aplicación heredada. En general, el modelado de datos ayuda a una organización a usar sus datos de manera efectiva para satisfacer las necesidades comerciales de información (Craig Stedman 2020).

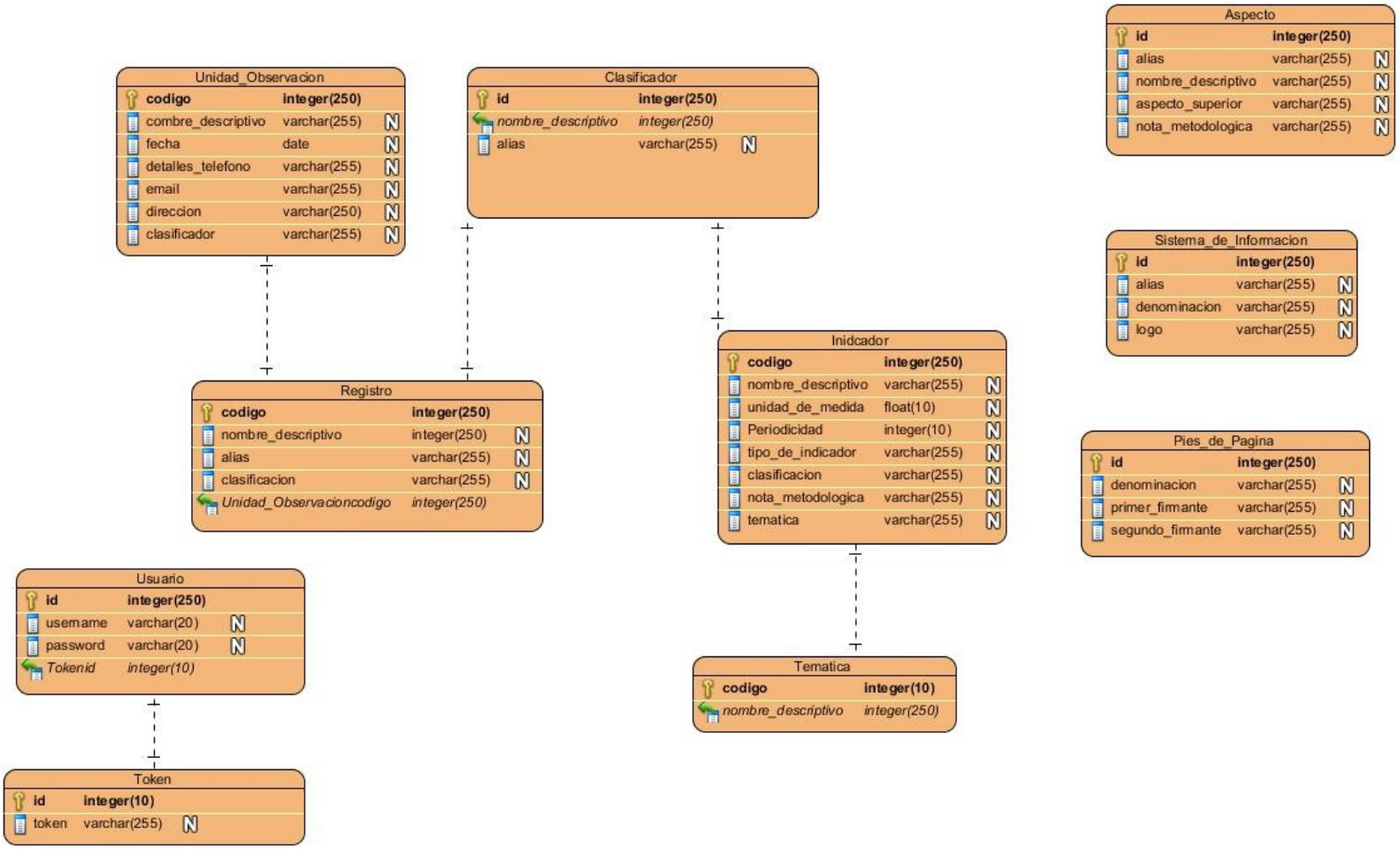


Figura 10. Modelo de la base de datos

Conclusiones

A partir del análisis y diseño del API de servicio del Módulo de Configuración se arribó a la siguiente consideración:

La elaboración de la propuesta de solución permitió construir el modelado del sistema, lo que posibilito conocer así las restricciones y necesidades del cliente. Se determinaron 17 requisitos funcionales y 3 no funcionales, los cuales mediante el escenario 4 de la metodología AUP para la UCI, posibilitó la obtención de las descripciones de los requisitos, a través de las historias de usuario, quedando detallada para su implementación posteriormente. El diseño de las clases mediante la utilización de los patrones GRASP y GOF posibilitó la obtención de un diseño que facilita el

mantenimiento de la propuesta de solución y se arribó a un diseño arquitectónico basado en REST, facilitando la integración con otros sistemas a través del protocolo HTTP.

CAPÍTULO 3: IMPLEMENTACIÓN Y EVALUACIÓN DEL API DE SERVICIO PARA EL MÓDULO DE CONFIGURACIÓN DE SIGE.

3.1 Diagrama de despliegue

Un diagrama de implementación en el lenguaje de modelado unificado modela la implementación física de los artefactos en los nodos. Muestra el hardware usado y los componentes instalados en el hardware, además de las conexiones físicas entre este y las relaciones entre componentes. Es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos (Acosta y Fernández 2014; SINGH 2012).

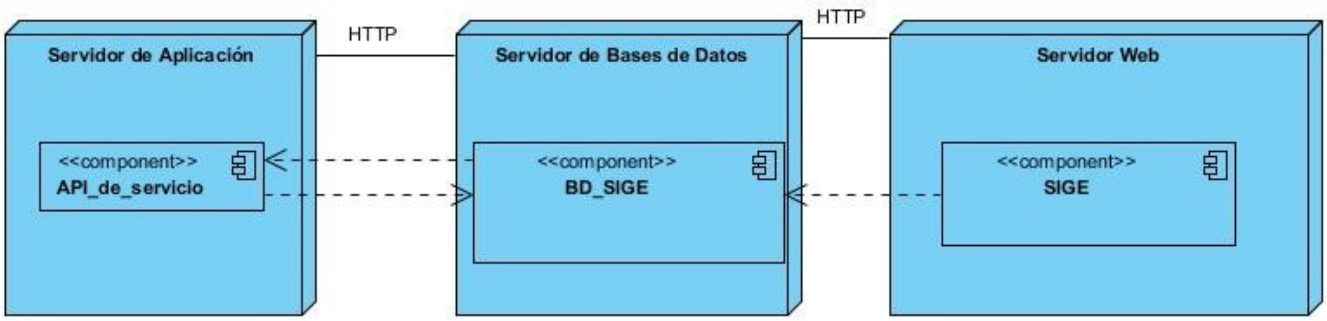


Figura 11. Diagrama de despliegue de la propuesta de solución.

**Descripción de los nodos del diagrama de despliegue **

- Servidor de Aplicación: contiene la API de servicio implementada.
- Servidor de Bases de datos: proporciona los servicios de la base de datos relacional BD_SIGE, donde están alojados todos los datos del módulo de configuración, datos como: unidad de observación, clasificador, registo, temática, aspecto, sistema de información, indicadores y pie de pagina.
- Servidor del Web: proporciona los paquetes necesarios de conexión en el modelo, así como funcionalidades que dan vida a elementos de base de datos del módulo de configuración.

3.2 Pruebas de software para la evaluación de la propuesta de solución

La prueba de software es un proceso, o una serie de procesos, diseñados para asegurarse de que el código de la computadora haga lo que fue diseñado para hacer y que no realice acciones no

deseadas o erróneas. El software debe ser predecible y consistente, y no ofrecer sorpresas a los usuarios. Las técnicas para encontrar problemas en un programa son variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad (Rodriguez Sanchez 2016).

A continuación, se describen los tipos de pruebas de software aplicadas, así como los métodos y técnicas empleadas para la evaluación de la propuesta de solución.

3.3 Aplicación de las pruebas de software

En el epígrafe se describen las pruebas de software realizadas en las disciplinas de pruebas internas propuestas en la metodología de desarrollo de software AUP para la UCI.

3.3.1 Pruebas internas

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas (Tamara 2015). En la disciplina se aplicaron pruebas unitarias mediante el método de caja blanca y la técnica de camino básico utilizando el producto de trabajo diseño de casos de pruebas. A continuación, se describen los pasos de la aplicación de esta prueba.

Pruebas unitarias

Las pruebas unitarias se centran en probar cada componente de código de un software de forma individual para asegurar que funcione de manera apropiada como unidad. Emplean técnicas de prueba que recorren caminos específicos en la estructura de control de los componentes (pruebas estructurales) (Juristo 2006). El método de prueba utilizado para la realización de esta prueba es Caja blanca y la técnica de prueba contenida en este método que se empleó fue la Técnica del camino básico.

Método de prueba: Caja blanca

El método de Caja blanca se enfoca en probar el sistema teniendo en cuenta la estructura interna del mismo. Verifica la correcta implementación de las unidades internas, las estructuras y sus relaciones y hacen énfasis en la reducción de errores internos (Juristo 2006).

Técnica de prueba: Camino básico

La técnica del Camino básico permite obtener una medida de la complejidad lógica de la codificación de software y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución independiente en un componente o programa. Un camino o ruta es una vía por la cual procede la ejecución a través de una función desde su inicio hasta el fin (Juristo 2006).

Pasos para la aplicación de la técnica del camino básico:

1. Dibujar el grafo de flujo de la funcionalidad o procedimiento a analizar

En la figura 11. se muestra la funcionalidad a analizar, detallando cada paso de implementación de inicio hasta fin, del método Autenticar usuario.

```
class login(ObtainAuthToken):

    def post(self, request, *args, **kwargs):
1------ login_serializer= self.serializer_class(data=request.data, context={'request': request})
2------ if login_serializer.is_valid():
3---------- user= login_serializer.validated_data['user']
4---------- if user.is_active:
5---------- token, created= Token.objects.get_or_create(user=user)
6---------- user_serializer= loginUserSerializer(user)
7---------- return Response({'token': token.key,'user': user_serializer.data,
                'message': 'inicio de sesion exitoso'},
                status= status.HTTP_201_CREATED)
            else:
8---------- return Response({'mesage': 'este usuario no esta permitido entrar' },
                status= status.HTTP_401_UNAUTHORIZED)
9
        else:
10 ---------return Response({'error': 'usuario o contraseña incorrecto'},
            status= status.HTTP_400_BAD_REQUEST)
11
```

Figura 11. Código de la implementación de Autenticar usuario.

En la figura 12. se realiza el diseño del flujo de grafo al procedimiento analizar, en este caso código de implementación de Autenticar usuario.

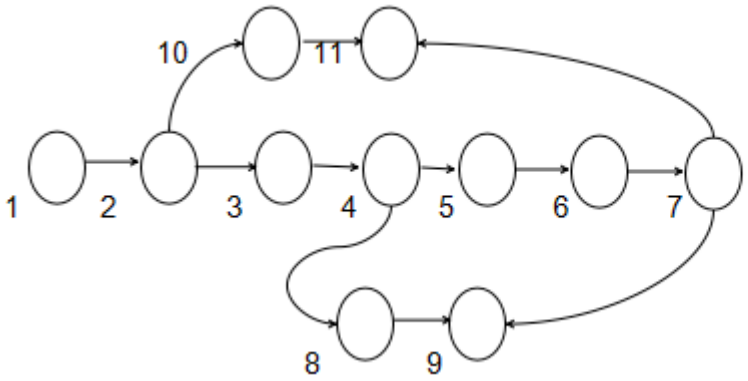


Figura 12. Grafo de flujo del código de la implementación del método Autenticar usuario.

2. Determinar la complejidad ciclomática del grafo de flujo resultante. La complejidad ciclomática del grafo (V(G)) se puede calcular de las tres formas siguientes:

- V(G) = A - N + 2 — A: cantidad de aristas

  V(G) = 12- 11 + 2= 3 — N: cantidad de nodos

- V(G) = P + 1 — P: número de nodos predicados

  V(G) = 2 + 1 = 3

- V(G) = Regiones (R) — R: regiones en el grafo

  = 3

3. Determinar el conjunto básico de caminos linealmente independientes

Camino básico 1: 1,2,3,4,5,6,7,9

Camino básico 2: 1,2,10,11

Camino 3: 1,2,3,4,8,9

4. Definir los casos de prueba para comprobar la ejecución de cada camino del conjunto básico

En el diseño de los casos de prueba se especifican los siguientes elementos:

- Descripción: contiene una descripción sobre las restricciones de los datos de entrada que debe tener el caso de prueba.
- Condición de ejecución: se especifican los parámetros que debe poseer el caso de prueba para que se cumpla una condición deseada como respuesta del funcionamiento del procedimiento.
- Entrada: se muestran los parámetros de entrada al procedimiento.
- Resultados esperados: se explica el resultado esperado de la ejecución del procedimiento.

Tabla 5. Diseño de casos de pruebas para el camino básico 1 del método Autenticar usuario

Diseño de caso de prueba para el camino 1	
Descripción	El proceso de autenticación inicia cuando el usuario desea autenticarse en la API, para ello debe introducir su usuario y contraseña correctamente. La aplicación debe reconocer al usuario en cuestión, a través de la consulta a la base de datos de usuarios permitidos para acceder a los recursos de la aplicación. Si el usuario correspondiente no tiene un inicio de sesión exitoso, se debe a dos razones fundamentales, escribió incorrectamente su usuario o contraseña, o el usuario no tiene los permisos necesarios para consumir la aplicación.
Condición de ejecución	El usuario que se registre tiene que ser válido, y además tener los permisos necesarios para consumir la aplicación.
Entrada	Usuario: enmanuelcc Contraseña: 12345678
Resultados esperados	Usuario autenticado en la aplicación, listo para consumir los recursos de la API.

A partir de la aplicación de los casos de pruebas se comprobó que el flujo de trabajo de las funcionalidades es correcto, ya que cada sentencia fue ejecutada al menos una vez, cumpliéndose así las condiciones de las pruebas.

3.3.2 Pruebas funcionales

Las pruebas funcionales tienen como objetivo ejercitar profundamente el sistema comprobando la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica. Se seleccionó este tipo de prueba con el objetivo de verificar y valorar las funcionalidades del sistema mediante la especificación de requisitos (Alarcos). El método de prueba utilizado para la realización de esta prueba es Caja negra y la técnica de prueba contenida en este método que se empleó fue la de Partición de equivalencia.

Método de prueba: Caja negra

Se enfoca en probar el sistema sin tomar en cuenta la estructura interna del mismo, su objetivo es validar que las salidas sean las esperadas. Se centra en encontrar las circunstancias en las que el sistema no se comporta conforme a las especificaciones establecidas (SEO, CHOI 2006).

Técnica de prueba: Partición de equivalencia

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada (Ecured).

Diseños de caso de pruebas

Es una parte de las pruebas de componentes y sistemas en las se diseñan los casos de prueba (entradas y salidas esperadas) para probar el sistema. Su objetivo es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requerimientos. Para su diseño, se selecciona una característica del sistema o componente que se está probando, un conjunto de entradas que ejecutan dicha característica, se documentan las salidas esperadas o rasgos de salida y donde sea posible se diseña una prueba automatizada que demuestre que las salidas reales y las esperadas son las mismas (Sommerville 2011). A continuación, se presenta el diseño de casos de prueba para el requisito funcional **RF 2.** Listar unidades de observación.

Abreviaturas utilizadas:

CP: Caso de Prueba.

HU: Historia de Usuario.

EC: Escenario

Tabla 6. Caso de Prueba Funcional del RF2 Listar unidades de observación

Caso de Prueba Funcional	
CP1_HU1	**HU1**: Listar unidades de observación.
Responsable: Enmanuel Contreras Camejo	
Descripción: El caso de prueba se inicia luego que el usuario se haya autenticado en la aplicación. Una vez autenticado debe especificar la URL que desea ver, en este caso “unidades”. Ya vez introducida la URL, se mostrará la información referente a las unidades de observación. URL: Ejemplo: /listar/unidades/	

Escenario	Descripción	Respuesta del sistema	Flujo del sistema
EC.1.1 Listar unidades de observación.	El usuario introduce la URL especifica de unidades de observación. Ejemplo: /listar/unidades/	Se muestra la información de las unidades de observación, además de un códogo de estado 200 OK.	El sistema lista las unidades de observación.
EC.1.2 Error cuando se introduce datos incorrectos	El usuario introduce la URL para listar unidades de observación, donde escribe la dirección incorrecta. Ejemplo:	Se mostrará un error de código 404 Not Found	El sistema detecta la URL incorrecta, donde devuelve un error de Pagina no encontrada.

	/listar/unidad/		

Para elaboración de la prueba anterior se utilizó la herramienta Postman para velar por el monitoreo del caso de prueba funcional del RF2. Listar unidad de observación. Ver figura 13 y 14.

En la figura 13 se muestra el resultado correcto del escenario 1.1 Listar unidad de observación, en la URL /listar/unidades/. El sistema devuelve el listado de unidades de observación, además de un código de éxito 200 OK.

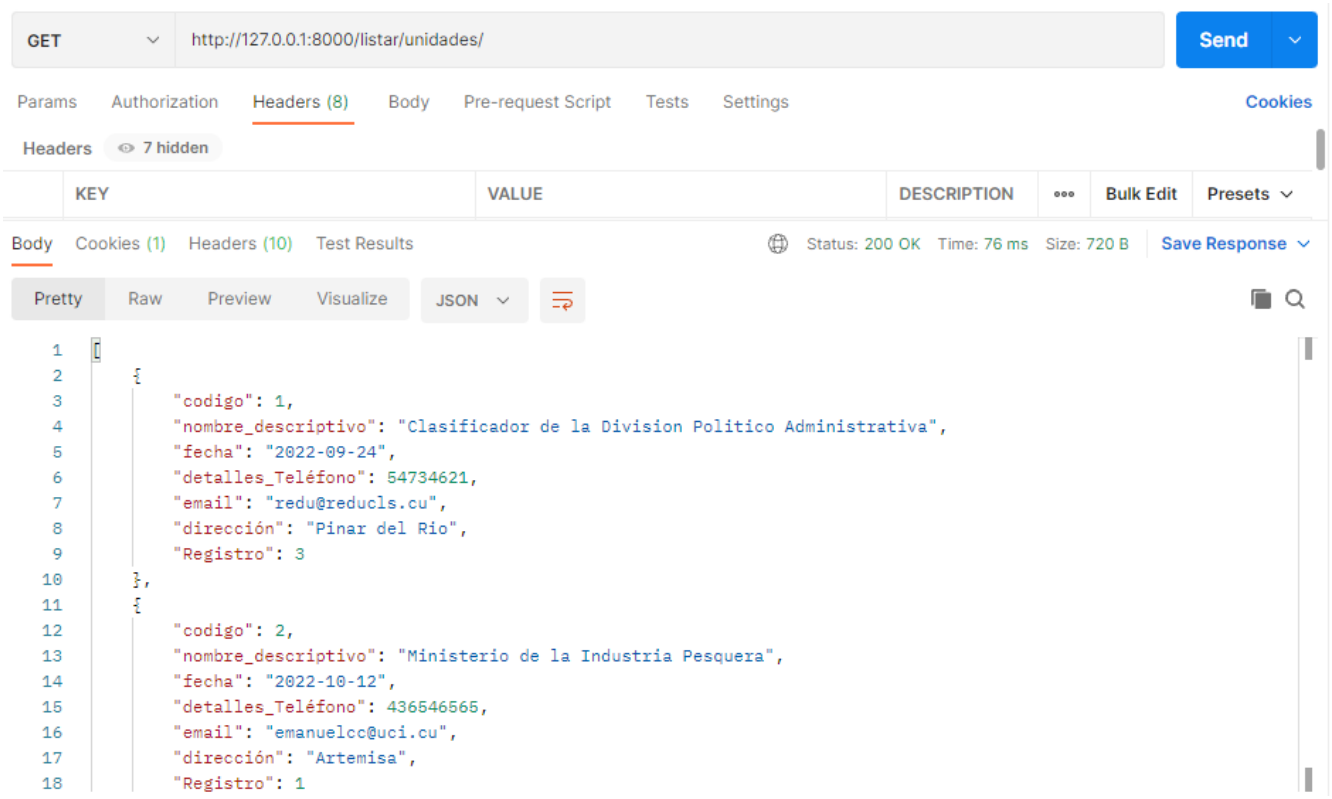


Figura 13. Caso de Prueba Funcional de la historia de usuario Listar unidad de observación. Escenario 1.1

En la figura 14 se muestra el resultado incorrecto del escenario 1.2 Listar unidades de observación, en la dirección URL listar/unidad/. Donde el sistema devuelve un código de error 404 Not Found, de página no encontrada.

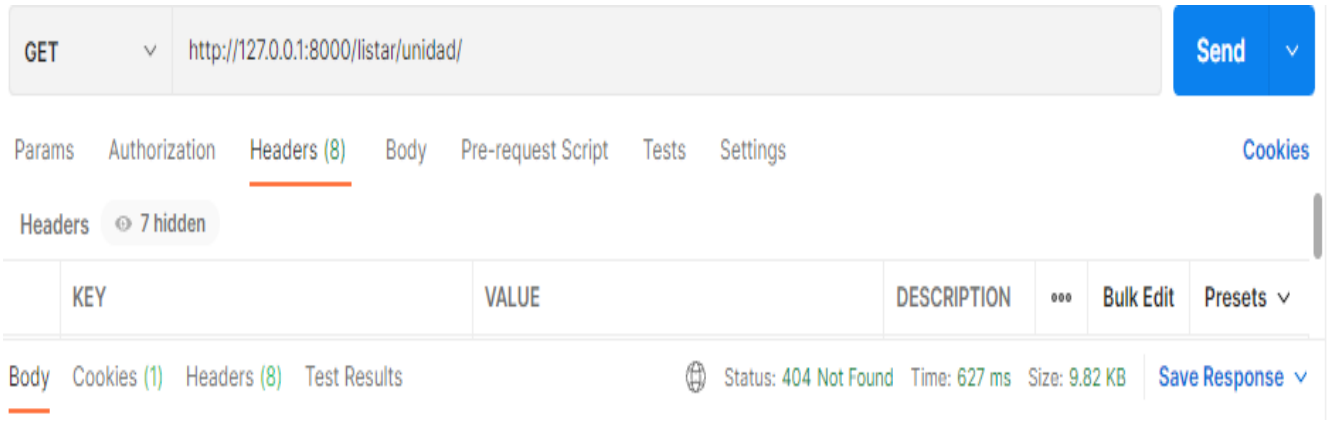


Figura 14. Caso de Prueba Funcional de la historia de usuario Listar unidad de observación. Escenario 1.2.

Conclusiones parciales

En este capítulo se realizó la construcción del Modelo de Despliegue, el cual describe cómo se encuentra distribuida física y lógicamente la arquitectura del sistema. Se aplicaron pruebas unitarias para comprobar el flujo de trabajo de las funcionalidades implementadas, demostrando que cada camino se ejecutó de forma satisfactoria. Mediante los casos de prueba se validó el software, verificando el correcto funcionamiento del sistema, comprobándose que no existían errores en las funciones operativas.

CONCLUSIONES GENERALES

El estudio realizado sobre el uso del módulo de configuración y así como la investigación referente a lo relacionado sobre las API, posibilitó determinar las funcionalidades necesarias para la propuesta de solución.

El análisis y diseño de la propuesta de solución permitió definir la estructura que iba a contener el sistema mediante la realización de los diagramas de clase del diseño y la utilización de los patrones arquitectónicos.

La implementación de la propuesta de solución permitió el desarrollo de la API que brinda información del módulo de configuración de SIGE, conteniendo como funcionalidades principales: listar unidades de observación, listar clasificadores, listar registros, listar aspectos, entre otros.

Las pruebas de software realizadas permitieron evaluar el funcionamiento de la API de servicio, la conformidad con los requisitos definidos y la eliminación de las no conformidades encontradas en las iteraciones, validando de esta forma el correcto funcionamiento de la propuesta de solución.

Recomendaciones

1. Se recomienda integrar la presente solución al Sistema Integrado para la Gestión Estadística, SIGE.

2. Se recomienda integrar las aplicaciones informáticas y sistemas web, a la API de servicio.

Referencias Bibliográficas

1. SIGE, 2014. Manual Gestión de Configuración. Consultado: 3 de mayo de 2022.

2. ONEI. Quienes Somos. *Oficina Nacional de Estadística e Información, Sitio en Actualización* [en línea]. Consultado: 4 de noviembre de 2022]. Disponible en: http://www.onei.gob.cu/node/1460

3. ONEI, 2017. ENCUESTA NACIONAL DE ENVEJECIMIENTO DE LA POBLACIÓN ENEP-2017. *Oficina Nacional de Estadística e Información, Sitio en Actualización* [en línea]. Consultado: 4 de noviembre de 2022]. Disponible en: http://www.onei.gob.cu/node/14725

4. ONEI, 2016. Encuesta Nacional sobre Igualdad de Género ENIG-2016 Informe de Resultados. *Oficina Nacional de Estadística e Información, Sitio en Actualización* [en línea]. Consultado: 4 de noviembre de 2022]. Disponible en: http://www.onei.gob.cu/node/14271

5. ONEI, 2012. Censo de Población y Viviendas 2012. *Oficina Nacional de Estadística e Información*, Sitio en Actualización [en línea]. Consultado: 4 de noviembre de 2022]. Disponible en: http://www.onei.gob.cu/node/13001

6. Quevedo, F, 2011. El proceso de observación y las variables. *Medwave* [en línea]. Consultado: 4 de novienbre de 2022. Disponible en:

7. INEGI. Instituto Nacional de Estadística y Geografía. *Principios básicos de las clasificaciones y recomendaciones para su elaboración* [en línea]. Consultado: 4 de noviembre de 2022]. Disponible en: https://www.inegi.org.mx/contenidos/productos/prod_serv/contenidos/espanol/bvinegi/productos/clasificadores/estad_recomen/702825062996.pdf

8. FAO. MÉTODOS DE RECOPILACIÓN DE DATOS. *Home | Food and Agriculture Organization of the United Nations* [en línea]. Consultado: 5 de enero de 2022]. Disponible en: https://www.fao.org/3/x2465s/x2465s08.htm

9. Henderson, Chloe, 2020. 4 tipos de integracion de sistemas- Pros frente a contras de cada método. *Intergrations Management System | AnyConnector* [en línea]. Consultado: 4 de mayo de 2022]. Disponible en: https://anyconnector.com/es/system-integration/types-of-system-integration.html

10. EDS Robotics, 2021. ¿Qué es la integración de sistemas? Ejemplos, Ventajas y Desventajas. *EDS Robotics* [en línea]. Consultado: 4 de mayo de 2022]. Disponible en: http://www.edsrobotics.com/blog/integracion-de-sistemas-que-es/

11. UCA. *Tecnologías para la Integración de Base de Datos en Web* [en línea]. Consultado el 4 de junio de 2022]. Disponible en:

12. Reyes, Carlos, 2020. ¿Qué es una API? - Academia Web. *Academia Web* [en línea]. Consultado: 4 de junio de 2022]. Disponible en: https://www.academiaweb.ca/que-es-una-api/

13. Fernández, Valcarcel, Alejandro, Gabriel y Santana, Ramírez, 2016. Repositorio Digital:API de servicios web en GDR 2.0 para la gestión de reportes desde diversos entornos de diseño. *Repositorio Digital:Página de inicio* [en línea]. Consultado: 4 de junio de 2022]. Disponible en: https://repositorio.uci.cu/jspui/handle/ident/9129

14. Kurtz, Jaime. ASP.NET MVC 4 and the Web AP. *Sergey Drozdov | Software Engineer* [en línea]. Consultado: 4 de noviembre de 2022]. Disponible en: http://sd.blackball.lv/library/ASP.NET_MVC_4_and_the_Web_API.pdf

15. Alodesk, 2020. API REST en 2020: ¿cómo puede beneficiar tu negocio? » Alodesk. *Alodesk* [en línea]. Consultado el 4 de mayo de 2022]. Disponible en: https://www.alodesk.cl/noticias/api-rest-2020/

16. Boeras, Mairelys [et al.]. Aplicando el método de Boehm y Turner. 6, La Habana : Ediciones Futuro, 2012, Vol. 5

17. Muños, Gonzáles, Dareyana de la Caridad, 2018. Repositorio Digital:API REST asociada al repositorio de Nova. *Repositorio Digital* [en línea]. Consultado: 4 de mayo de 2022]. Disponible en: https://repositorio.uci.cu/jspui/handle/123456789/9923

18. Eriksson, Hans-Erik and Penker, Magnus, 1998. "UML Toolkit" Wiley Computer Publishing

19. Grady, Booch, James, Rumbaugh e Ivar Jacobson, 2006. El Lenguaje Unificado de Modelado, 2.º Edición. Madrid, Pearson Addison Wesley. ISBN 978-84-782-9076-5

20. Priscila Landeros, Gómez Ruth. Herramienta CASE. *Monografias* [en línea]. Consultado: 4 de mayo de 2022]. Disponible en: https://www.monografias.com/trabajos14/herramicase/herramicase

21. Pressman, Roger S, 2002. Ingeniería de Software, un enfoque práctico. Quinta edición. S.l. : McGraw-Hill Companies. ISBN: 8448132149.

22. Marín, Rafael, 2019. Los gestores de bases de datos (SGBD) más usados. *Canal Informática y TICS* [en línea]. Publicado: 16 de abril de 2019, consultado el 4 de mayo de 2022]. Disponible en: https://www.inesem.es/revistadigital/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/

23. López Herrera, Patricia, 2016. Comparación del desempeño de los Sistemas Gestores de Bases de Datos MySQL y PostgreSQL.

24. Capterra. *IDE (entorno de desarrollo integrado)* [en línea]. Consultado: 21 de mayo de 2021. Disponible en:

25. Juganaru Mathieu, Mihaela 2014. Introducción a la programación. Grupo Editorial Patria. ISBN 978-607-438-920-3. Consultado: 21 de mayo de 2022. Disponible en: https://editorialpatria.com.mx/pdffiles/9786074384154.pdf/

26. Python Software Foundation. El tutorial de Python — documentación de Python - 3.11.0. *3.11.0 Documentation* [en línea]. Consultado: 4 de mayo de 2022. Disponible en: https://docs.python.org/es/3/tutorial/

27. Docts.django. FAQ: General - Documentation- Django. *Django* [en línea]. Consultado: 4 de junio de 2022. Disponible en: https://docs.djangoproject.com/en/4.1/faq/general/

28. Damián de Luca. Visual Studio Code - Características principales. *Desarrollo Web Capacitación y Consultoría* [en línea]. Consultado: 4 de junio de 2022]. Disponible en: https://www.ecured.cu/Visual_Studio_Code

29. Prasad Acharya, Durga, 2022. Las 10 mejores herramientas de desarrollo y prueba de API. *GeekFlare* [en línea]. Consultado: 30 de octubre de 2022. Disponible en: Postman. Postman API Platform. *Postman* [en línea]. Consultado: 30 de octubre de 2022]. Disponible en: https://www.postman.com/

30. Molina Hernández, Yenisel, Granda Dihigo, Ailec, & Velázquez Cintra, Alionuska, 2019. Los requisitos no funcionales de software. *Una estrategia para su desarrollo en el Centro de Informática*

Médica. Revista Cubana de Ciencias Informáticas, 13(2), 77-90 [en línea]. Consultado: 2 de noviembre de 2022. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992019000200077&lng=es&tlng=pt.

31. Jeffries, R., Anderson, A., Hendrickson, C, 2001. "Extreme Programming Installed". Addison-Wesley. ISBN: 0201708426

32. Garlan, D. y Shaw, M, 1993. “An introduction to software architecture”. Advances in Software Engineering and Knowledge Engineering, 11-39.

33. Craig, Larman, 1999. UML Y PATRONES. Introducción al análisis y diseño orientado a objetos. s.l. : PRENTICE HAL.

34. Sixto. “¿Patrones?”, 2003

35. Fabbri, Serena. *Design Patterns* [en linea]. Consultado: 23 de septiembre de 2022. Disponible en: https://studylib.net/doc/9737765/design-patterns

36. Craig Stedman. *Data modeling* [en linea]. Consultado: 23 de septiembre de 2022. Disponible en : https://www.techtarget.com/searchdatamanagement/definition/data-modeling

37. Acosta, Tòmas y Hernàndez, Nodelvis. *Módulo de reportes webmétricos para el motor de búsqueda Orión. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas* [en línea]. Consultado: 16 de octubre de 2022. Disponoble en: https://repositorio_institucional.uci.cu/jspui/handle/ident/8296.

38. Singh, Sanjay Kumar; Singh, Amarjeet, 2012. Software testing. Vandana Publications.

39. Rodriguez Sanchez, T. *Metodología de desarrollo para la Actividad productiva de la UCI.Universidad de las Ciencias Informáticas*. La Habana. Cuba [en línea]. Consultado: 16 de octubre de 2022. Disponible en: https://repositorio_institucional.uci.cu/jspui/handle/ident/8917.

40. Tamara, R. S, 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.*

41. Juristo N, M. A, (17 de octubre de 2006). *Tecnicas de evaluacion de software*

42. Alarcos. Pruebas del Sistema. *Directriz: Pruebas del Sistema* [en línea]. Consultado: 6 de septiembre de 2022. Disponible en:

https://alarcos.esi.uclm.es/ipsw/metrica3/metrica_3/guidances/guidelines/pruebas_sistema_E169CC06.html

43. SEO, Kwang Ik; CHOI, Eun Man, 2006. Comparison of five black-box testing methods for object-oriented software. *En Fourth International Conference on Software Engineering Research, Management and Applications* (SERA'06). IEEE. p. 213-220.

44. Ecured. *Pruebas de caja negra- Ecured* [en línea]. Consultado: 16 de octubre de 2022. Disponible en: https://www.ecured.cu/Pruebas_de_caja_negra#Partici.C3.B3n_equivalente

45. Sommerville, Ian, 2011. Ingeneria del Software Novena ed. España . España : s.n. Disponible en:

Anexos

Anexo 1: Entrevista realizada a especialistas de SIGE

Objetivo: Obtener información sobre los datos generados por el módulo de configuración, así como requisitos funcionales y no funcionales de la propuesta de solución.

1. ¿ Qué datos genera el módulo de configuración y qué relaciones tiene cada uno de ellos?

2. ¿Cuáles son los aspectos fundamentales a tener en cuenta cuando se consume la API de servicio?

3. ¿Cree usted necesario la creación de la API que permita realizar estos servicios? ¿Por qué?

4. ¿Cuáles servicios tendría este servicio?

Anexos 2: Historias de Usuario

Historia de Usuario para RF: 1

Autenticar usuario.

Tabla 7. Historia de usuario del RF1 Autenticar Usuario.

Historia de Usuario	
Número: 3	**Nombre del requisito:** Autenticar usuario.
Programador: Emanuel Contrera	**Iteración Asignada:** 1 iteración
Prioridad: Alta	**Tiempo Estimado:** 2 semanas
Riesgo de Desarrollo: No aplica	**Tiempo Real:** 2 semanas
Descripción: El proceso de autenticación inicia cuando el usuario desea autenticarse en el sistema, para ello debe introducir su usuario y contraseña correctamente. El sistema debe reconocer al usuario en cuestión, a través de la consulta a la base de datos de usuarios permitidos para acceder a los recursos de la aplicación. Una vez autenticado el sistema devuelve un token de autenticación donde podrá consumir los servicios de la aplicación.	
Observaciones: ● Para acceder a la URL debe utilizar el método POST.	

● La URL para acceder a la funcionalidad está compuesta por: La dirección donde esté publicado. Ejemplo: /login/ ● Proceso de autenticación: Usuario: Contraseña:

Historia de Usuario para RF: 4

Mostrar listado de clasificadores estadísticos

Tabla 8. Historia de usuario del RF4 Listar clasificadores.

Historia de Usuario	
Número: 4	**Nombre del requisito:** Listar clasificadores estadísticos
Programador: Emanuel Contrera	**Iteración Asignada:** 1 iteración
Prioridad: Alta	**Tiempo Estimado:** 2 semanas
Riesgo de Desarrollo: No aplica	**Tiempo Real:** 2 semanas
Descripción: La aplicación muestra un listado de clasificaciones estadísticas con sus diferentes atributos. Listados de forma ascendente por su número de código. Las clasificaciones presentaran atributos como: código, nombre descriptivo, alias y dominio.	
Observaciones: ● El usuario debe autenticarse. ● Para acceder a la URL debe utilizar el método GET.	

● La URL para acceder a la funcionalidad está compuesta por: La dirección donde esté publicado el servicio Ejemplo: /listar/clasificaciones/

Historia de Usuario para RF: 5

Buscar clasificadores estadísticos.

Tabla 9. Historia de usuario del RF5 Buscar clasificadores estadísticos.

Historia de Usuario	
Número: 5	**Nombre del requisito:** Buscar clasificadores estadísticos.
Programador: Emanuel Contrera	**Iteración Asignada:** 1 iteración
Prioridad: Alta	**Tiempo Estimado:** 2 semanas
Riesgo de Desarrollo: No aplica	**Tiempo Real:** 2 semanas
Descripción: La aplicación muestra los clasificadores o clasificador, del conjunto de listado de acuerdo al criterio de búsqueda. El criterio de búsqueda estará regida por el nombre descriptivo del clasificador seleccionado. Una vez encontrado la especificación de la búsqueda; la aplicación muestra los clasificadores o clasificador con sus diferentes atributos: código, nombre descriptivo, registro, estado, provincia y municipio.	
Observaciones: ● El usuario debe autenticarse. ● Para acceder a la URL debe utilizar el método GET. ● La URL para acceder a la funcionalidad está compuesta por: La dirección donde esté publicado el servicio.	

Ejemplo: /listar/clasificaciones/?search= clasificador de actividades/

Historia de Usuario para RF: 6

Mostrar listado de registros estadísticos.

Tabla 10. Historia de usuario del RF6 Listar registros estadísticos.

Historia de Usuario	
Número: 6	**Nombre del requisito:** Listar registros estadísticos.
Programador: Emanuel Contrera	**Iteración Asignada:** 1 iteración
Prioridad: Alta	**Tiempo Estimado:** 2 semanas
Riesgo de Desarrollo: No aplica	**Tiempo Real:** 2 semanas
Descripción: La aplicación muestra un listado de registros estadísticos con sus diferentes atributos. Listados de forma ascendente por su número de código. Los registros presentarán atributos como: código, alias y nombre descriptivo.	
Observaciones: ● El usuario debe autenticarse. ● Para acceder a la URL debe utilizar el método GET. ● La URL para acceder a la funcionalidad está compuesta por: La dirección donde esté publicado el servicio Ejemplo: /listar/registros/	

Historia de Usuario para RF: 7

Buscar registros estadísticos.

Tabla 11. Historia de usuario del RF7 Buscar registros estadísticos.

Historia de Usuario	
Número: 7	**Nombre del requisito:** Buscar registros estadísticos.
Programador: Emanuel Contrera	**Iteración Asignada:** 1 iteración
Prioridad: Alta	**Tiempo Estimado:** 2 semanas
Riesgo de Desarrollo: No aplica	**Tiempo Real:** 2 semanas
Descripción: La aplicación muestra los registros o registro, del conjunto de listado de acuerdo al criterio de búsqueda. El criterio de búsqueda estará regida por el nombre descriptivo del registro seleccionado. Una vez encontrado la especificación de la búsqueda; la aplicación muestra los registros o registro con sus diferentes atributos: código, nombre descriptivo, registro, estado, provincia y municipio.	
Observaciones: ● El usuario debe autenticarse. ● Para acceder a la URL debe utilizar el método GET. ● La URL para acceder a la funcionalidad está compuesta por: La dirección donde esté publicado el servicio. Ejemplo: /listar/registros/?search= registro de empresas mixtas/	

Anexo 3 Diseño de caso de prueba

Buscar unidades de observación

Abreviaturas utilizadas:

CP: Caso de Prueba.

HU: Historia de Usuario.

EC: Escenario.

Tabla 12. Caso de Prueba Funcional del RF3 Buscar unidades de observación.

Caso de Prueba Funcional			
CP2_HU3		**HU2**: Buscar unidades de observación.	
Responsable: Enmanuel Contreras Camejo			
Descripción: El caso de prueba se inicia luego que el usuario se haya autenticado en la aplicación. Una vez autenticado debe escoger el endpoint que define la URL que desea ver, en este caso “unidades”, además de “ ?search” que especifica lo que se quiere buscar, y en este caso el nombre descriptivo que identifica al recurso que quiere seleccionar. Ya una vez escogida la URL, se mostrará la información referente a la unidad de observación. Dirección de la URL: Ejemplo: /listar/unidades/?search= Empresas Mixtas/			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo del sistema**
EC.1 Buscar unidades de observación.	El usuario selecciona la URL especifica de unidades de observación. Ejemplo: /listar/unidades/?search= Empresas	Muestra la información de las unidades de observación, además de un código de éxito 200 OK.	El sistema lista los clasificadores correspondientes a la búsqueda realizada.
EC.1.2 Error cuando se introduce datos incorrectos.	El usuario introduce la URL, donde escribe la dirección con datos incorrectos. Ejemplo: /listar/unidades/?= empresas	Se mostrará un error de código 404 Not Found.	El sistema detecta la dirección URL incorrecta, donde devuelve un error de Pagina no encontrada.

Listar clasificadores estadísticos

Abreviaturas utilizadas:

CP: Caso de Prueba.

HU: Historia de Usuario.

EC: Escenario.

Tabla 13. Caso de Prueba Funcional del RF4 Listar de clasificadores estadísticos.

Caso de Prueba Funcional			
CP3_HU4	**HU4**: Listar clasificadores estadísticos.		
Responsable: Enmanuel Contreras Camejo			
Descripción: El caso de prueba se inicia luego que el usuario se haya autenticado en la aplicación. Una vez autenticado debe escoger el endpoint que define la URL que desea ver, en este caso “clasificadores”. Ya una vez escogida la URL, se mostrara la información referente a las clasificaciones estadísticas. Dirección de la URL: Ejemplo: /listar/clasificadores/			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo del sistema**
EC.1 Listar clasificadores.	El usuario selecciona la URL especifica de clasificadores. Ejemplo: listar/clasificadores/	Muestra la información de los clasificadores, además de un código de éxito 200 OK.	El sistema lista los clasificadores estadísticos.
EC.1.2 Error cuando se introduce datos	El usuario introduce la URL, donde escribe la dirección	Se mostrará un error de código 404 Not Found.	El sistema detecta la dirección URL incorrecta, donde devuelve un error

incorrectos.	con datos incorrectos Ejemplo: /listar/clasificador/		de Pagina no encontrada.

Buscar clasificadores estadísticos.

Abreviaturas utilizadas:

CP: Caso de Prueba.

HU: Historia de Usuario.

EC: Escenario.

Tabla 14. Caso de Prueba Funcional del RF5 Buscar clasificadores estadísticos.

Caso de Prueba Funcional			
CP4_HU5	**HU5**: Buscar clasificadores.		
Responsable: Enmanuel Contreras Camejo			
Descripción: El caso de prueba se inicia luego que el usuario se haya autenticado en la aplicación. Una vez autenticado debe escoger el endpoint que define la URL que desea ver, en este caso “clasificadores”, además de “ ?search” que especifica lo que se quiere buscar, y en este caso el nombre descriptivo que identifica al recurso que quiere seleccionar. Ya una vez escogida la URL, se mostrará la información referente a la clasificadores. Dirección de la URL: Ejemplo: /listar/clasificaciones/?search= clasificador de actividades/			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo del sistema**
EC.1 Buscar clasificadores estadísticos.	El usuario selecciona la URL	Muestra la información de los clasificadores,	El sistema lista los clasificadores correspondientes a la

	especifica. Ejemplo: / listar/clasificaciones /? search=clasificador	además de un código de éxito 200 OK.	búsqueda realizada.
EC.1.2 Error cuando se introduce datos incorrectos.	El usuario introduce la URL, donde escribe la dirección con datos incorrectos Ejemplo: / listar/clasificadores/ serach= nomenclador	Se mostrará un error de código 404 Not Found.	El sistema detecta la dirección URL incorrecta, donde devuelve un error de Página no encontrada.

Listar registros estadísticos

Abreviaturas utilizadas:

CP: Caso de Prueba.

HU: Historia de Usuario.

EC: Escenario.

Tabla 15. Caso de Prueba Funcional del RF6 Listar registros estadísticos.

Caso de Prueba Funcional	
CP5_HU6	**HU6**: Listar registros.
Responsable: Enmanuel Contreras Camejo	

Descripción: El caso de prueba se inicia luego que el usuario se haya autenticado en la aplicación. Una vez autenticado debe escoger el endpoint que define la URL que desea ver, en este caso “registros”. Ya una vez escogida la URL, se mostrará la información referente a los registros estadísticos. Dirección de la URL: Ejemplo: /listar/registros/			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo del sistema**
EC.1 Listar registros.	El usuario selecciona la URL especifica de registros. Ejemplo: /listar/registros/	Muestra la información de los registros estadísticos.	El sistema lista los registros estadísticos.
EC.1.2 Error cuando se introduce datos incorrectos.	El usuario introduce la URL, donde escribe la dirección con datos incorrectos Ejemplo: /listar/registro/	Se mostrará un error de código 404 Not Found.	El sistema detecta la dirección URL incorrecta, donde devuelve un error de Página no encontrada.

Buscar registros estadísticos.

Abreviaturas utilizadas:

CP: Caso de Prueba.

HU: Historia de Usuario.

EC: Escenario.

Tabla 16. Caso de Prueba Funcional del RF7 Buscar registros estadísticos.

Caso de Prueba Funcional			
CP6_HU7		**HU7**: Buscar registros estadísticos.	
Responsable: Enmanuel Contreras Camejo			
Descripción: El caso de prueba se inicia luego que el usuario se haya autenticado en la aplicación. Una vez autenticado debe escoger el endpoint que define la URL que desea ver, en este caso “registros”, además de “ ?search” que especifica lo que se quiere buscar, y en este caso el nombre descriptivo que identifica al recurso que quiere seleccionar. Ya una vez escogida la URL, se mostrará la información referente del registro o registros. Dirección de la URL: Ejemplo: /listar/registros/?search= Empresas Mixtas/			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo del sistema**
EC.1 Buscar registros.	El usuario selecciona la URL especifica. Ejemplo: /listar/registros/?search=empresas	Muestra el listado de registros, además de un código de exito 200 OK.	El sistema lista los registros correspondientes a la busqueda realizada.
EC.1.2 Error cuando se introduce datos incorrectos	El usuario introduce la URL, donde escribe la dirección con datos incorrectos Ejemplo: /listar/registros/search=empresas	Se mostrará un error de código 404 Not Found	El sistema detecta la dirección URL incorrecta, donde devuelve un error de Pagina no encontrada.