ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DISEÑO E IMPLEMENTACIÓN DE UNA CADENA DE PROCESAMIENTO DE DATOS QUE INTEGRE UN MECANISMO ROBUSTO DE MANEJO DE DOCUMENTOS"

INFORME DE PROYECTO INTEGRADOR

Previo a la obtención del Título de:

INGENIERO EN COMPUTACIÓN

Bosco Armando Andrade Bravo

Kevin Omar Palacios Alvarez

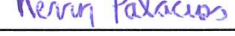
GUAYAQUIL – ECUADOR

AÑO: 2019

DECLARACIÓN EXPRESA

"Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Bosco Andrade Bravo y Kevin Palacios Alvarez damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Bosco Andrade Bravo

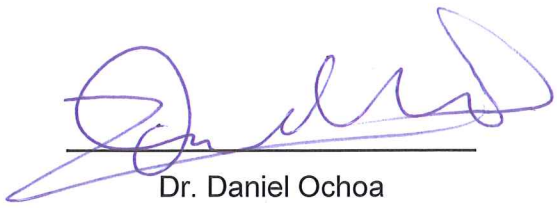


Kevin Palacios Alvarez

EVALUADORES

Dr. Boris Vintimilla

PROFESOR DE LA MATERIA



Dr. Daniel Ochoa

PROFESOR TUTOR

RESUMEN

En Ecuador, el principal problema de las plataformas que comparten datos públicos acerca de clima, salud o transporte, es que los archivos publicados no contienen metadatos. Para encontrar información de una ciudad hay que visitar varios sitios web, descargar, ordenar y catalogar la información de forma manual. Además, la información recuperada no posee un historial de los cambios que se han efectuado sobre ella.

Para resolver este problema, este proyecto convirtió el sistema Data City, propiedad de la Escuela Superior Politécnica del Litoral, en un proveedor y visualizador de datos con un mecanismo robusto de manejo de documentos. Este proyecto implementó una arquitectura que permitió la integración de diferentes fuentes de datos heterogéneos y el registro de todos los estados de los conjuntos de datos. Además, mantuvo operativa la funcionalidad de graficación de mapas y series de tiempo que incluía la versión original del sistema Data City. Es importante mencionar que uno de los algoritmos de graficación de mapas, fue optimizado con el objetivo de disminuir el tiempo de graficación.

CKAN fue la herramienta que se integró al sistema Data City para la gestión de documentos. Debido a esto, se logró reducir el tiempo promedio de búsquedas aproximadamente 19%. Además, se logró la integración de nuevas fuentes de datos como el sistema City Health Map, que provee datos sobre incidencias de salud en la ciudad de Guayaquil al sistema Data City. Por otro lado, la nueva arquitectura de Data City permitió la recolección y almacenamiento de documentos con cualquier formato. Finalmente, los cambios que se realizaron sobre uno de los algoritmos de graficación de mapas, lograron una reducción del tiempo de graficación de aproximadamente 96%.

Palabras clave: Data City, CKAN, Datos Abiertos, Documentos, Científicos.

ABSTRACT

In Ecuador, the main problem with platforms that share public data about weather, health or transport is that the published documents do not contain metadata. To find information about a city you must visit several websites, download, sort and catalog the information manually. In addition, the retrieved information does not have a history of the changes that have been made to it.

To solve this problem, this project converted the Data City system, owned by the Escuela Superior Politécnica del Litoral, in a data provider and viewer with a robust document management mechanism. This project implemented an architecture that allowed the integration of different heterogeneous data sources and the recording of the whole states of the data sets. In addition, it kept operational the map and time series graphing functionality that was included on the original version of the Data City system. It is important to mention that one of the map graphing algorithms was optimized to decrease the graphing time.

CKAN was the tool integrated into the Data City system for document management. Due to this, the average search time was reduced by approximately 19%. In addition, the integration of new data sources was achieved, such as the City Health Map system, which provides data on health incidents in the city of Guayaquil to the Data City system. On the other hand, Data City's new architecture allowed the collection and storage of documents in any format. Finally, the changes made to one of the mapping algorithms resulted in a reduction in mapping time of approximately 96%.

Keywords: Data City, CKAN, Open Data, Documents.

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
RECLIMA	Resiliencia Climática
GAD	Gobierno Autónomo Descentralizado
TMS	Tile Map Server
TSS	Time Series Server
HTTP	Hypertext Transfer Protocol

SIMBOLOGÍA

S	Segundos
MB	Megabytes

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. Introducción

Este capítulo describe la descripción del problema, objetivos asociados y justificación de este proyecto. Estos tópicos involucran una breve delineación de los sistemas ResClima y Data City, haciendo énfasis en sus funcionalidades, objetivos y la relación con el desarrollo de este trabajo. Paralelamente, se realizó una investigación que involucró el análisis de proyectos de código abierto dedicados a la gestión de documentos, con el objetivo de identificar los componentes que podrían ser utilizados en la solución.

1.1. Descripción del problema

La Escuela Superior Politécnica del Litoral (ESPOL), junto al Gobierno Autónomo Descentralizado del cantón Durán (GAD Durán), trabajaron en el proyecto de investigación Resiliencia Climática (ResClima) [1]. Este sistema, permite la adquisición, visualización y análisis de datos de la ciudad de Durán. ResClima fue diseñado específicamente para la recolección de datos climáticos. Como consecuencia, este sistema se encuentra limitado al almacenamiento, lectura y visualización de datos relacionados a temperatura, presión barométrica, precipitación por detección, humedad, series de tiempo, capas vectoriales y ráster.

Posteriormente al desarrollo de ResClima, se construyó una extensión que permite la creación y manipulación de tableros interactivos para sumarizar la información relevante de ResClima. A este sistema se lo denominó Data City, y de ahora en adelante, usaremos este nombre para referirnos al sistema original y sus extensiones. El objetivo de Data City es proveer a los ciudadanos, autoridades e investigadores, datos públicos crudos y procesados que sirvan como plataforma para las futuras iniciativas de ESPOL en el ámbito de las ciudades inteligentes.

En Ecuador no existen portales con el enfoque de Data City, sin embargo, hay varias plataformas que publican datos y algunas implementan

mecanismos estándares para su intercambio. Por ejemplo, INIAP (Instituto Nacional de Investigaciones Agropecuarias de Ecuador) o SIPA (Sistema de Información Pública Agropecuaria) comparten datos públicos agropecuarios de Ecuador como estimaciones de superficies, indicadores sectoriales, estadísticas agro-productivas, etc. Por otro lado, existen también portales como el del GAD de Quito, que proporciona información de interés social como salud, seguridad, hábitat y vivienda [2]. Sin embargo, el principal problema radica en que los archivos publicados no contienen metadatos, por lo tanto, su búsqueda se dificulta. Además, la información recuperada no posee un formato adecuado para su posterior procesamiento, por ejemplo, es común encontrar tablas en formato PDF o mapas en formato PNG. Para encontrar información de una ciudad hay que visitar varios sitios webs, descargar, ordenar y catalogar la información de forma manual para que sea realmente útil.

Este proyecto propone convertir la versión actual de Data City en un proveedor y visualizador de datos que se integre a un sistema robusto de manejo de documentos. El enfoque de este proyecto es diseñar una arquitectura que permita implementar cadenas de recolección, procesamiento y visualización de datos que mantenga registros de todos los estados de procesamiento de los conjuntos de datos. Además, se busca soportar la integración de fuentes de datos externas o propias.

1.2. Justificación

Este proyecto busca fomentar la cultura de compartir y difundir información con el objetivo de reforzar diversas áreas del conocimiento [3], a través de la provisión de información fácilmente accesible para cualquier individuo de la sociedad.

1.3. Objetivos

1.3.1. Objetivo General

Implementar una cadena de procesamiento de datos que integre un mecanismo robusto de manejo de documentos que soporte la recolección, almacenamiento e indexación de datos procedentes de diferentes fuentes para convertir el sistema Data City en un proveedor y visualizador de datos.

1.3.2. Objetivos Específicos

- Implementar una arquitectura que provea un mecanismo robusto de manejo de
- documentos.
- Desarrollar una funcionalidad que soporte el ingreso de documentos con cualquier extensión al sistema Data City.
- Integrar la versión actual de Data City a un sistema de gestión de documentos.
- Reducir el tiempo promedio de graficación de mapas en el visualizador Data City.

1.4. Marco Teórico

El concepto de Ciudad Inteligente se define como la inteligencia urbana que permite a las personas tener la comodidad y seguridad necesaria en su diario vivir [4]. Las ciudades inteligentes están equipadas con métodos de recolección masiva de datos, que son generados a través de sensores, instituciones, departamentos, etc. También métodos para transmitir estos datos a los sistemas de soporte de decisiones, cuyo objetivo es mejorar aspectos del sector urbano como calidad de vida, tránsito, salud, etc. [5].

La gran variedad de fuentes de datos existentes origina que la información se encuentre dispersa en la red. Por lo tanto, surge la necesidad de implementar herramientas que concentren todos los datos en un solo lugar y de forma ordenada, con el objetivo de compartir, procesar y reutilizar la

información [6]. Estas herramientas son conocidas como plataformas de administración de datos, y una de las más usadas en el mundo es CKAN [7].

CKAN es un portal de código abierto que se encarga de gestionar y almacenar conjuntos de datos. Además, provee una base de datos dedicada, que soporta el almacenamiento de información estructurada y cualquier tipo de documentos; también provee una API que permite integrar nuevos clientes [8].

Un portal web popular que utiliza CKAN es data.gov [9], desarrollado y administrado por la Administración de Servicios Generales de Estados Unidos. Este portal recopila datos de agricultura, clima, educación, etc. Además, se rige con el esquema de datos abiertos del proyecto Open Data [10], que comprende una serie de campos obligatorios como etiquetas, título, descripción, última actualización, editor, en todos los conjuntos de datos almacenados. A continuación, se describen los módulos más importantes que provee CKAN.

1.4.1. Conjunto de datos

En el marco de trabajo CKAN, un conjunto de datos es un paquete de recursos de diferente formato cuyo contenido está relacionado a un tema específico. Por ejemplo, las estadísticas de delincuencia de un país o reportes de clima de una ciudad. Un conjunto de datos posee dos características principales:

a. Meta data

Información sobre los datos almacenados en un conjunto de datos. Por ejemplo: título, descripción, fecha de publicación, etc.

b. Recursos

Documentos que conforman el conjunto de datos. Pueden ser de cualquier tipo, un conjunto de datos puede tener tantos recursos como desee.

1.4.2. API

CKAN provee una API de tipo RPC que permite utilizar todas las funcionalidades que posee CKAN desde sistemas externos. Esta API está desarrollada en Python y sus servicios principales soportan la administración de organizaciones, usuarios, grupos, conjuntos de datos, recursos, etc.

1.4.3. File Store

CKAN provee una funcionalidad llamada File Store que consiste en una base de datos de tipo PostgreSQL y permite a los usuarios crear conjuntos de datos y recursos. Por defecto, todos los documentos ingresados a través del File Store son almacenados en la misma máquina en la que se encuentra instalado CKAN [11]. Sin embargo, existen varias extensiones que permiten almacenar los conjuntos de datos y recursos en diferentes servicios de almacenamiento en la nube como AWS S3 o Azure.

1.4.4. Apache Solr

Proyecto de código abierto escrito en Java y basado en un software de Apache llamado Lucene [12], [13]. Debido a que Solr permite realizar búsquedas sobre grandes cantidades de datos, CKAN utiliza esta herramienta como motor de búsqueda. Esto implica que todas las solicitudes de búsqueda que ingresan a CKAN a través de su API, son gestionadas por Apache Solr.

1.4.5. Extensiones

CKAN provee múltiples extensiones que permiten realizar tareas adicionales que no están incluidas en el código base de CKAN. Estas extensiones son desarrolladas por la comunidad activa encargada del mantenimiento de CKAN utilizando Python como lenguaje de programación.

CAPÍTULO 2

2. Metodología

Este capítulo describe de forma general la arquitectura de cada componente de la nueva versión del sistema Data City y las modificaciones que serán ejecutadas sobre cada uno de ellos, tomando en consideración los objetivos generales y específicos, y herramientas investigadas en el capítulo 1.

2.1. Nueva arquitectura de Data City

Para la creación de la versión 2 del sistema Data City, se definió una arquitectura de componentes escalable y mantenible. La Figura 2.1 muestra los cuatro componentes principales que conforman el sistema Data City v2: Aplicación Web, CKAN, Tile Map Server y Time Series Server. La arquitectura de los componentes mencionados anteriormente es descrita en subsecciones posteriores de este capítulo.

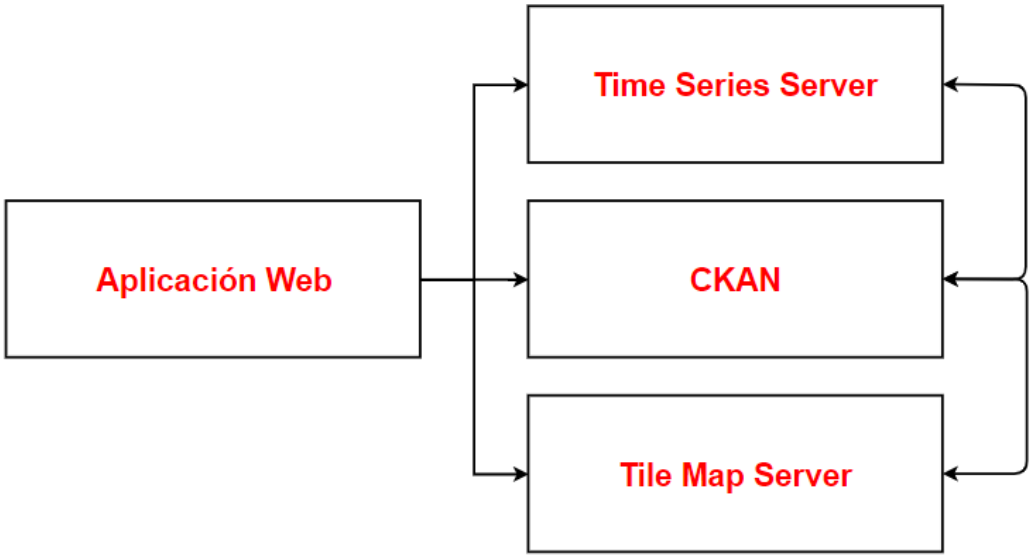


Figura 2.1 Arquitectura general de Data City v2. Fuente: elaboración propia.

2.2. Aplicación Web

La Figura 2.2 muestra la arquitectura de la nueva aplicación web de Data City v2 junto a sus tres módulos principales: administración de documentos y series de tiempo, graficación y búsquedas. En este punto, es importante aclarar qué, el término ***documentos*** hace referencia a documentos de cualquier tipo, incluyendo capas ráster y vectoriales. Las series de tiempo

no son consideradas documentos debido a que son mediciones recolectadas desde sensores meteorológicos y almacenadas en una base de datos.

El módulo administración de documentos y series de tiempo, posee funcionalidades relacionadas a la creación, descarga, edición, actualización y eliminación de documentos y series de tiempo. Por otro lado, el módulo graficación posee funcionalidades relacionadas a la graficación de mapas con capas ráster y vectoriales y, además, la graficación de series de tiempo. Finalmente, el módulo búsquedas, es el encargado de realizar búsquedas de documentos y series de tiempo, utilizando diversos parámetros.

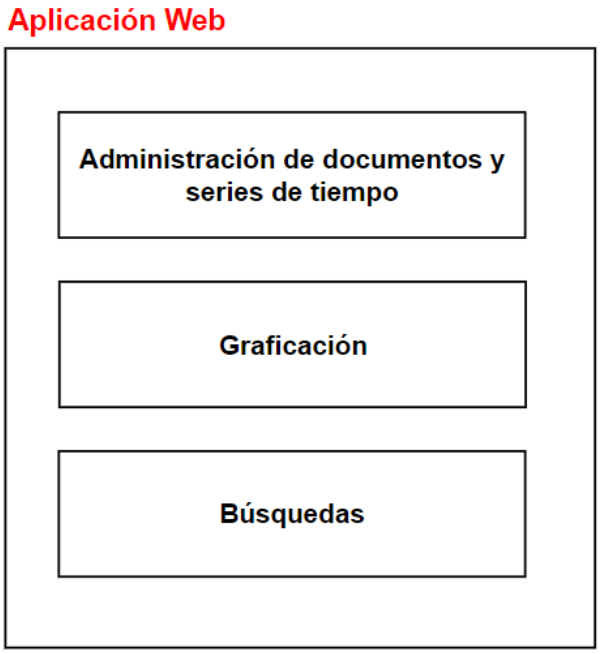


Figura 2.2 Módulos de la aplicación web de Data City v2.

Fuente: elaboración propia.

Sobre cada uno de estos módulos fueron realizadas modificaciones que permitieron soportar la arquitectura descrita en la Figura 2.1 y los objetivos planteados en el capítulo 1. Estos cambios involucraron que las funcionalidades de los módulos redirijan sus solicitudes a componentes externos como CKAN, Tile Map Server o Time Map Sever, como se muestra en la Tabla 2.1.

Tabla 2.1 Relación entre funcionalidades y componentes de Data City v2

Fuente: elaboración propia.

Módulo	Funcionalidad	Componente
Administración de documentos y series de tiempo	Administración de capas ráster	CKAN
	Administración de capas vectoriales	CKAN
	Administración de series de tiempo	Time Series Server
	Administración de documentos	CKAN
Graficación	Graficación de capas ráster	Tile Map Server
	Graficación de capas vectoriales	CKAN
	Graficación de series de tiempo	Time Series Server
Búsquedas	Búsquedas	CKAN

2.3. CKAN

Con el objetivo de proveer al sistema Data City un mecanismo robusto de manejo de documentos, se integró el portal de datos CKAN. En la arquitectura mostrada en la Figura 2.3, se evidencian sus herramientas como el API, o el motor de búsqueda Apache Solr. La funcionalidad principal de CKAN es administrar todos los documentos ingresados en Data City v2. Debido a esto, el resto de los componentes mostrados en la Figura 2.1, deben relacionarse con CKAN para garantizar el correcto funcionamiento de Data City.

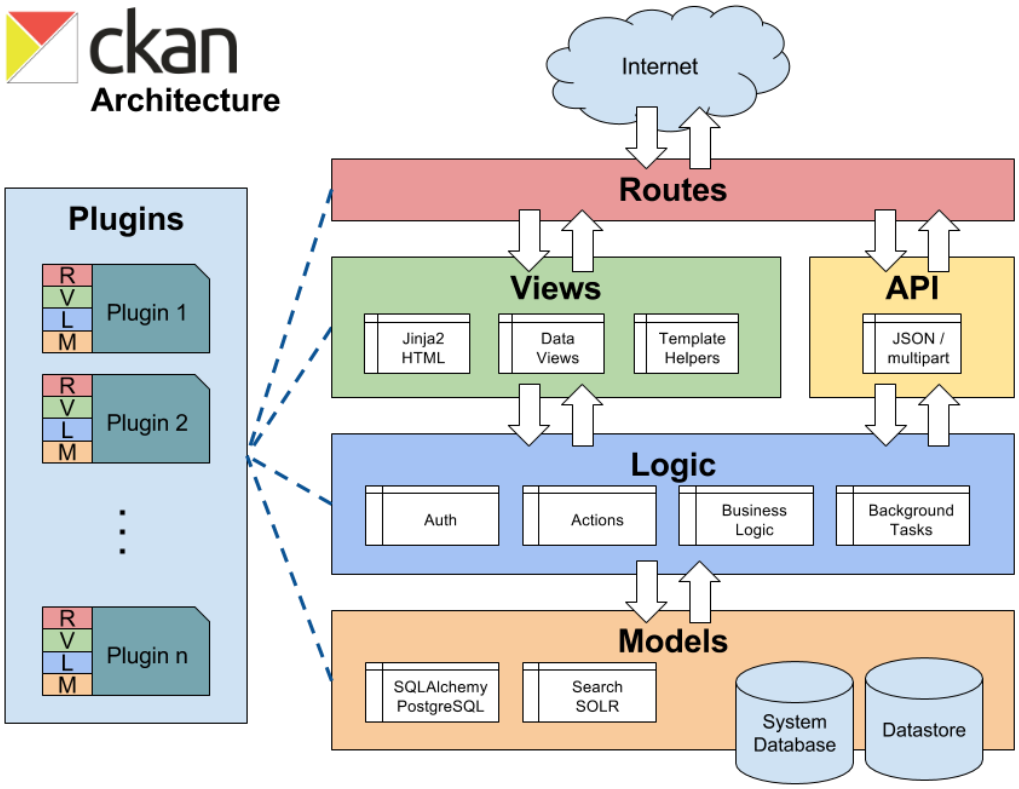


Figura 2.3 Arquitectura de CKAN [14].

2.4. Tile Map Server

En el sistema Data City v1, la funcionalidad para graficación de capas ráster consistía en un algoritmo TMS que recibía dos parámetros: un nivel de acercamiento y el identificador de la capa solicitada. Con este identificador se realizaba una consulta a la base de datos y se obtenía la ruta de almacenamiento de la capa solicitada; luego el algoritmo leía esta capa y realizaba el correspondiente procesamiento en base al nivel de acercamiento recibido con el objetivo de convertir la capa en una imagen con formato png y retornarla a la aplicación web para realizar la graficación del mapa.

El componente Tile Map Server (TMS) surge luego de definir una arquitectura basada en componentes para Data City v2 y la necesidad de reducir el tiempo de graficación de mapas con capas ráster. Su creación fue basada en la funcionalidad para graficación de capas ráster de Data City v1. Sin embargo, se realizaron las modificaciones descritas a continuación.

Para optimizar los tiempos de respuesta, se crearon tres instancias de TMS en un mismo servidor, orquestadas por el balanceador de carga NGINX [15], como se evidencia en la Figura 2.4. Además, se complementó esta configuración con un mecanismo de cacheo de imágenes. De esta forma cuando se recibe una solicitud, el balanceador de carga primero busca la imagen en la cache; si la encuentra, la retorna inmediatamente. Caso contrario, la solicitud es enviada a uno de los servidores TMS para que realice el procesamiento de la capa solicitada y retorne la imagen correspondiente.

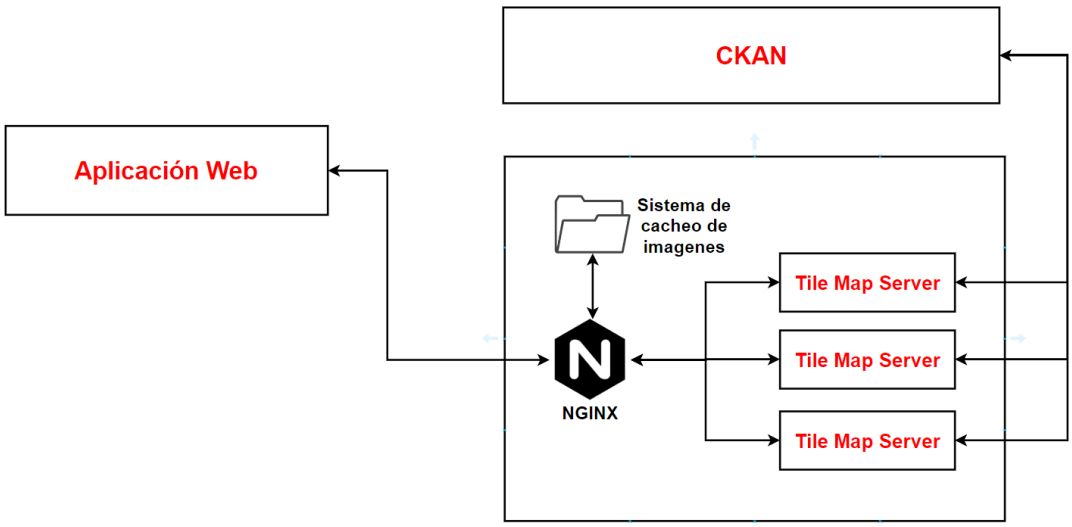


Figura 2.4 Arquitectura del componente Tile Map Server.
Fuente: elaboración propia.

Por otro lado, debido a que en Data City v2, CKAN es el componente encargado de la administración de documentos, fue necesario que TMS se relacionara directamente con CKAN para poder acceder a la ruta de almacenamiento de las capas que han sido solicitadas.

2.5. Time Series Server

El componente Time Series Server surge luego de definir una arquitectura basada en componentes para Data City v2. Para la creación de este componente, se incluyó las funcionalidades de administración y graficación de series de tiempo que ya estaban implementadas en la aplicación web de

Data City v1 y cuyo objetivo principal era obtener mediciones provenientes de sensores meteorológicos, para su posterior procesamiento y visualización como series de tiempo.

Dicho esto, la administración y graficación de series de tiempo en Data City v2, posee la misma metodología de trabajo que en Data City v1 con la única diferencia que el almacenamiento se realizó sobre el sistema de base de datos de CKAN, mostrado en la Figura 2.3, con el modelo de base de datos utilizado en Data City v1 y mostrado en la Figura 2.5.

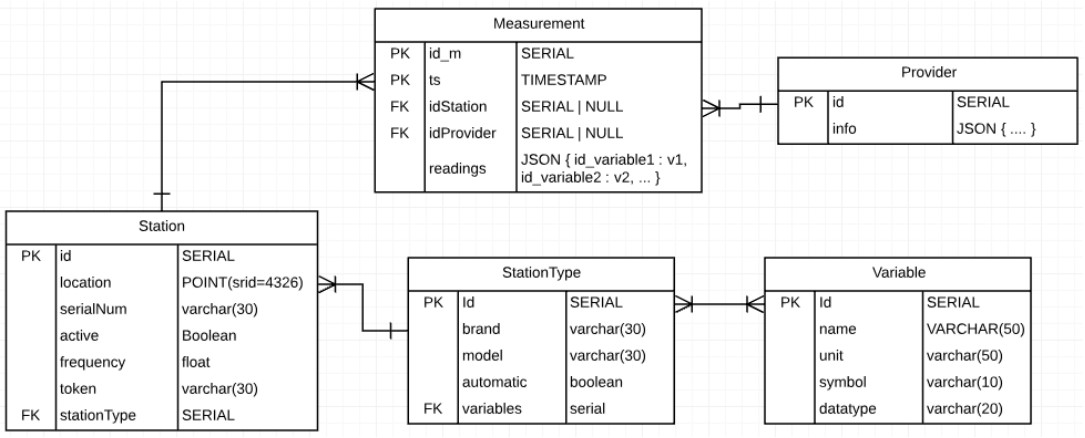


Figura 2.5 Diagrama de base de datos del componente Time Series Server. Fuente: documentación del proyecto Data City v1.

CAPÍTULO 3

3. Implementación de la solución

Este capítulo describe todos los procedimientos realizados sobre cada uno de los componentes que conforman el proyecto Data City v2 para soportar la metodología propuesta en el capítulo 2.

3.1. Aplicación Web

3.1.1. Administración de documentos y series de tiempo

La aplicación web de Data City v1 integra operaciones para crear, descargar, editar, actualizar y remover documentos relacionados únicamente a series de tiempo, capas vectoriales y ráster, como se muestra en la Figura 3.1.

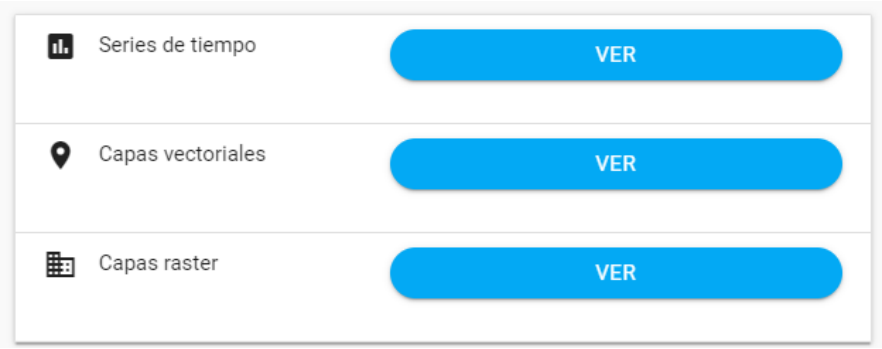


Figura 3.1 Documentos soportados por Data City v1. Fuente: elaboración propia.

Por otro lado, con el objetivo de que Data City v2 soporte las mismas operaciones descritas anteriormente, pero para ***cualquier*** tipo de documento, ***sin importar su extensión***, se creó una sección adicional denominada “Documentos”, como se muestra en la Figura 3.B2.

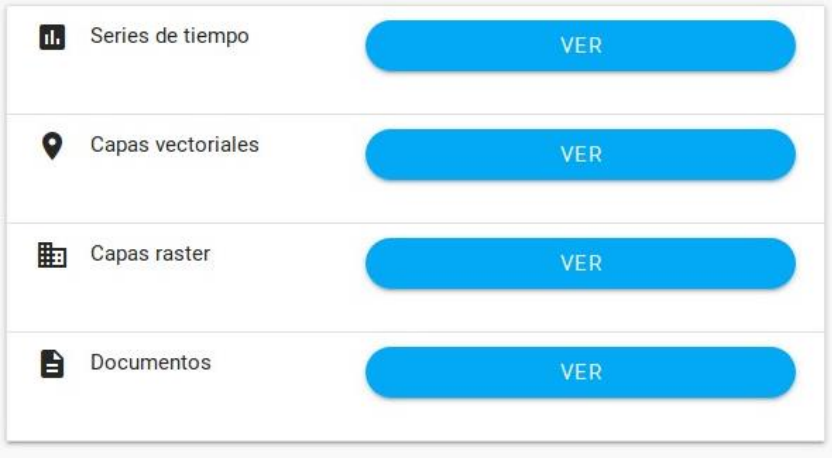


Figura 3.2 Nueva funcionalidad para administrar documentos. Fuente: elaboración propia.

La integración de CKAN como portal de datos, originó que las tres secciones mostradas en la Figura 3.2 quedaron obsoletas, por lo que tuvieron que ser modificadas para coexistir junto a CKAN. De igual forma, la nueva sección denominada “Documentos”, tuvo que ser implementada teniendo en consideración la existencia de CKAN.

Dicho esto, los procesos de lectura y escritura para documentos fueron redirigidos directamente a CKAN a través de su API, utilizando los servicios listados en la Tabla 3.1. Por otro lado, los procesos de lectura y escritura para las series de tiempo tuvieron que ser redirigidos al componente Time Series Server. En la sección 3.3, se explica detalladamente su implementación.

Tabla 3.1 Servicios del API de CKAN utilizados para la administración de documentos. Fuente: elaboración propia.

Servicio	Descripción
package_create	Crea un nuevo conjunto de datos
package_update	Actualiza un conjunto de datos
package_delete	Borra un conjunto de datos

package_search	Busca un conjunto de datos
package_show	Muestra información de un conjunto de datos
resource_create	Crea un nuevo recurso
resource_update	Actualiza un recurso
resource_delete	Borra un recurso
resource_show	Muestra información de un recurso

La inserción de cualquier documento desde la aplicación web, desencadena la ejecución de los siguientes pasos:

1. Creación de un ***conjunto de datos*** cuya meta data se asocie a la meta data ingresada a través de la aplicación web.
2. Inserción del documento ingresado, en el conjunto de datos creado en el punto 1. Es decir, el documento ingresado a través de la aplicación web, se convierte en un ***recurso*** del conjunto de datos credo en el punto 1.

3.1.2. Graficación

Como se describe en la Tabla 2.1, el módulo de graficación de la aplicación web de Data City v2 está compuesto de tres funcionalidades: graficación de capas ráster, graficación de capas vectoriales y graficación de series de tiempo. Debido a que la graficación de capas ráster y series de tiempo, utiliza los componentes Tile Map Server y Time Series Server, respectivamente, la implementación de estas funcionalidades se describe en las secciones 3.3 y 3.4. Por otro lado, la graficación de capas vectoriales únicamente se relaciona con el componente CKAN, por lo que a continuación se describen las modificaciones realizadas sobre esta funcionalidad.

Para la graficación de capas vectoriales en la aplicación web de Data City v1, se enviaba el identificador de la capa que se quería graficar a la funcionalidad de administración de capas vectoriales. Esta funcionalidad realizaba una consulta a la base de datos para obtener la ruta de almacenamiento de la capa solicitada, que era almacenada como un documento con formato GeoJSON en la máquina en la que estaba instalado Data City v1. Posteriormente, se extraían los datos del GeoJSON respectivo y se retornaba el contenido para que sea graficado.

Debido a que en Data City v2 todos los documentos se encuentran almacenados en CKAN como recursos, la implementación de esta funcionalidad para Data City v2 solicita la información de la capa al componente CKAN a través de su API. La solicitud se realiza enviando el identificador del recurso al servicio resource_show con el cual se obtiene toda la información del recurso, incluyendo un enlace mediante el cual se solicita el documento a CKAN a través de una solicitud http. Posteriormente, de la misma manera que en Data City v1 se retorna el contenido del documento a la funcionalidad de graficación de capas vectoriales.

3.1.3. Búsquedas

La integración de CKAN como gestor de archivos, originó que el módulo de búsquedas de la aplicación web de Data City v1 quedara obsoleto. Para resolver este problema y realizar búsquedas sobre los conjuntos de datos existentes, se utilizó el servicio package_search [16] del API de CKAN. Los parámetros de búsqueda ingresados por el usuario a través de la aplicación web, deben ser relacionados con la meta data que poseen los conjuntos de datos de CKAN, como se muestra en la Tabla 2.1.

Tabla 3.2 Relación entre parámetros de búsqueda y metadatos de conjuntos de datos. Fuente: elaboración propia.

Parámetro de búsqueda ingresados en la aplicación web	Meta data del conjunto de datos de CKAN
Texto	title | notes
Categorías	Tags
Fecha	metadata_created
Tipo	dataset-type
Geoespacial	Spatial

3.2. CKAN

En esta sección se detallaron los procedimientos implementados para la instalación de CKAN y las extensiones complementarias. De las tres modalidades de instalación que tiene CKAN [15], se utilizó la instalación desde el código fuente con el objetivo de tener un mejor control sobre sus funcionalidades y más flexibilidad al momento de agregar extensiones.

La instalación se llevó a cabo siguiendo detalladamente los pasos descritos en el tutorial de instalación para la modalidad elegida [17]; la versión utilizada fue v2.8. Es importante mencionar que, aunque el tutorial está escrito para las versiones 14.04 y 16.04 de Ubuntu, en este proyecto se utilizó la versión 18.4 sin ningún inconveniente.

Paralelamente a esta instalación se creó un archivo de instalación que contenía todos los pasos del tutorial, con el objetivo de automatizar los procesos de instalación.

3.2.1. Extensión para búsquedas geoespaciales

Para que CKAN soporte parámetros de búsquedas geoespaciales sobre los conjuntos de datos, fue necesario instalar la extensión ckanext-spatial [18] a través del tutorial de instalación descrito en la página web de la extensión [19]. Además, para que esta extensión funcione

correctamente, se necesitó que los conjuntos de datos tuviesen el campo *spatial* en su lista de meta data. El valor de este campo debe estar en formato GeoJSON y representa el área geográfica relacionada al conjunto de datos, en forma de puntos o polígonos.

3.3. Time Series Server

En Data City v2, el componente Time Series Server (TSS) es el encargado de la administración y graficación de series de tiempo. TSS fue escrita en Python, bajo el marco de trabajo Django 1.11, utilizando las funcionalidades de administración y graficación de series de tiempo ya implementadas en Data City v1.

Inicialmente, estas funcionalidades manejaban una base de datos propia para el almacenamiento de datos. Sin embargo, luego de la creación del componente TSS, estas funcionalidades tuvieron que ser ligeramente modificadas con el objetivo de almacenar todos los datos en la base de datos del componente CKAN. Para lograr esta integración, fue necesario agregar las tablas descritas en la Figura 2.5 a la base de datos de CKAN. Esto fue posible debido a que ambas bases de datos son de tipo PostreSQL [20], por lo que no surgieron problemas al momento de la integración.

Es importante mencionar que debido a que TSS no obtiene los datos a través de la lectura de documentos sino a través de lecturas a tablas de base de datos, no existe una interacción con la API de CKAN, sino con su base de datos directamente.

3.4. Tile Map Server

El componente Tile Map Server (TMS) fue implementado en Python sobre el marco de trabajo Django 1.11. y debe ser instalado sobre la misma maquina donde se encuentra el componente CKAN para que puede acceder directamente a los recursos que se encuentran almacenados en Data City v2.

Para realizar la integración entre TMS y CKAN, fue necesario realizar modificaciones en la lógica del algoritmo del componente TMS. Debido a que en Data City v2, todos los documentos son almacenados como un recurso de CKAN, el algoritmo del componente TMS fue modificado para recibir el identificador del recurso (capa ráster) que se desea graficar.

A diferencia de la implementación en Data City v1, el algoritmo del componente TMS ya no realiza consultas a la base de datos para obtener la ruta de almacenamiento de la capa, sino que accede directamente a la ruta donde está almacenada la capa. Esto es posible debido a que la ruta total del recurso se puede obtener a través del valor de la variable ckan.storage_path definida en el archivo de configuración de CKAN, que hace referencia a la ruta del directorio raíz en el que todos los documentos serán almacenados, junto a información extraída desde el identificador del recurso. La Figura 3.3 describe cómo se compone el identificador de un recurso.

Figura 3.3 Estructura del identificador de un recurso en CKAN.
Fuente: elaboración propia.

CKAN almacena los recursos en el disco duro utilizando una jerarquía de directorios de dos niveles. Tomando esto como referencia, y sabiendo que en la variable ckan.storage_path está definida la ruta del directorio raíz en la que se almacenan los recursos, para el identificador de ejemplo de la Figura 3.3, la ruta de almacenamiento es:

<ckan.storage_path>/276/a0c/57-fc2e-405b-a421-3297a591d2e4

Una vez obtenida la ruta, el algoritmo TMS realiza el procedimiento subsecuente.

3.4.1. Balanceo de carga

Con el objetivo de gestionar paralelamente múltiples solicitudes provenientes de la aplicación web, en Data City v2 se crearon tres instancias del componente TMS. Estas instancias fueron gestionadas por el balanceador de carga NGINX. Esto implica que todas las solicitudes que ingresan son tramitadas en primera instancia por NGINX, utilizando el algoritmo least-connected cuya funcionalidad es identificar el componente TMS con menos carga de trabajo para luego redirigir la solicitud a ese componente específico.

3.4.2. Sistema de cacheo de imágenes

El sistema de cacheo de imágenes fue implementado con el objetivo de reducir los tiempos de respuesta del componente TMS. Este sistema se implementó utilizando la funcionalidad de cacheo que posee la herramienta NGINX. Para habilitarlo, se creó un archivo de configuración en el que se especificó que todas las solicitudes entrantes con el punto final **/tmp/** sean analizadas. Además, se definió que el tiempo máximo que estas imágenes estarán almacenadas en cache sea 60 minutos y que la jerarquía de directorios en la que se almacenarán todas las imágenes sea de dos niveles.

Las configuraciones descritas anteriormente involucran que, al recibir una solicitud, el sistema de cacheo primero verifique si esa solicitud específica ya ha sido requerida dentro de los 60 minutos previos. Si la respuesta es positiva, se retorna la imagen solicitada inmediatamente; caso contrario, la solicitud es redirigida a una de las instancias del componente TMS a través del balanceador de carga.

3.4.3. Optimización de algoritmo TMS

Finalmente se realizó una optimización en el algoritmo del componente TMS; específicamente en el manejo de estilos de las capas ráster. En la implementación de este algoritmo en Data City v1, en la etapa de procesamiento que correspondía a la aplicación de estilos a la capa, se aplicaba el estilo a toda la capa y luego se realizaba un recorte en base

al nivel de zoom recibido, aplicar el estilo a toda la capa provocaba que se incremente el tiempo de procesamiento, y por lo tanto el incremento del tiempo de respuesta al cliente. La optimización consistió en realizar primero el recorte y luego aplicar los estilos, de esta forma el estilo es aplicado a una porción más pequeña de la capa, lo que disminuye el tiempo de procesamiento.

3.5. Gestión de usuarios y organizaciones

La aplicación web de Data City v1 fue implementada utilizando el marco de trabajo Django. Esta herramienta posee un completo sistema de autenticación que gestiona cuentas de usuarios, grupos y permisos. Por otro lado, en el proyecto Data City v2 aparece CKAN. Este componente maneja un sistema de autorización basado en organizaciones [21]. Cada usuario de CKAN debe pertenecer a una organización para poder crear conjuntos de datos y recursos.

Por los motivos antes descritos, la aplicación web de Data City v2 debe relacionar el sistema de gestión de usuario de Django con el sistema de gestión de usuario de CKAN. El objetivo es que un usuario de la aplicación web de Data City se relacione directamente con un usuario de CKAN. Esto fue posible luego de agregar al modelo de usuarios de la aplicación web, los campos mostrados en la Tabla 2.2.

Tabla 3.3 Campos adicionales agregados al modelo de usuario.
Fuente: elaboración propia.

Campo	Descripción
ckan_api_key	Clave necesaria para utilizar el API de CKAN, es generada por CKAN al momento de crear un usuario.
ckan_user_id	Identificador de usuario dentro de CKAN
ckan_organization_id	Identificador de la organización a la que pertenece el usuario

3.6. Integración de nuevas fuentes de datos

La nueva arquitectura que se implementó sobre el proyecto Data City v2, soporta la integración de nuevas fuentes de datos a través del componente CKAN. Cualquier documento que sea ingresado por una fuente externa a Data City, puede ser mostrado en los resultados de búsqueda y descargado desde la aplicación web, siempre que contenga la meta data descrita en la Tabla 2.3, tanto en el conjunto de datos, como en los recursos ingresados. Así mismo, cualquier capa ráster o capa vectorial que sea ingresada por fuentes externas puede ser graficada si contiene la meta data descrita en la Tabla 2.3.

Tabla 3.4 Metadatos necesarios para soportar conjuntos de datos provenientes de fuentes externas. Fuente: elaboración propia.

	NOMBRE DE CAMPO	DESCRIPCIÓN	VALOR
METADATA CONJUNTO DE DATOS	dataset-type	Tipo de conjunto de datos	raster | vector | document
	source	Nombre de fuente de datos	'Hospital León Becerra'
	tag_string	Lista de categorias	'Salud, Clima'
	spatial	-	-
	notes	Descripción del conjunto de datos	-
METADATA DE RECURSO	format	Extensión del recurso	json | pdf | zip | csv | etc.
	url_type	-	upload

CAPÍTULO 4

4. Análisis de resultados

Este capítulo describe los resultados que se obtuvieron después de la implementación de la solución propuesta en este proyecto. Aquí se detallan los resultados obtenidos de la integración de nuevas fuentes de datos. Además, debido a las mejoras realizadas en el componente Tile Map Server, se realizaron pruebas para analizar los resultados de esta optimización. Finalmente, con el objetivo de analizar el comportamiento de la nueva arquitectura ante grandes cantidades de datos, se realizaron pruebas sobre el componente de búsqueda. Estas pruebas fueron realizadas sobre Data City v1 y Data City v2.

4.1. Integración con nuevas fuentes de datos

Con la nueva arquitectura implementada, se logró que dos nuevas fuentes de datos se incorporen a Data City v2. La primera fue una aplicación web que recopila y visualiza datos de salud de la ciudad de Guayaquil [22]. Este portal ingresa documentos de capas vectoriales a Data City v2, con la meta data descrita en la Tabla 3.4. De esta forma, las capas subidas por este sistema pudieron ser listadas y graficadas en Data City v2.

La otra fuente de datos es una aplicación web que recopila resultados de encuestas realizadas a través de una aplicación móvil. Los resultados son estructurados en un documento con formato CSV y enviados a Data City v2. De esta forma, estos documentos pudieron ser listados en los resultados de búsqueda de la aplicación web de Data City v2.

Con la integración de estas nuevas fuentes de datos, se puede comprobar que la arquitectura planteada en este proyecto permite la integración de cualquier fuente de datos externa a Data City v2, utilizando correctamente la meta data descrita en la Tabla 3.4.

4.2. Comparativa entre Data City v1 y Data City v2

En esta sección se realiza una comparación de los módulos de búsqueda y graficación de capas ráster y vectoriales de Data City v1 y Data City v2.

4.2.1. Configuración de ambientes

El punto inicial de este proceso fue la creación y configuración de ambientes para la ejecución de pruebas. Se crearon dos máquinas virtuales en la plataforma Oracle Cloud bajo las características descritas en la Tabla 4.1. En la primera máquina se instaló Data City v1 y en la segunda, Data City v2 con la nueva arquitectura implementada en este proyecto.

Tabla 4.1 Características de las máquinas virtuales utilizadas para la ejecución de pruebas. Fuente: elaboración propia.

Tipo	CPUs	RAM	Almacenamiento	Ancho de Banda
VM.Standard2.2	2	30 GB	46.6 GB	2 Gbps

4.2.2. Ingreso de datos

Luego de la creación de los ambientes descritos en el punto 4.2.1, fue necesario añadir datos para poder ejecutar las pruebas. Para llevar a cabo este paso, se ingresó la misma cantidad de datos tanto en Data City v1 como en Data City v2. Al inicio del proyecto se recibió un conjunto de datos de prueba que consistía en 31 capas ráster y 38 capas vectoriales, recopilando en total 69 capas diferentes con tamaño entre 1 Mb y 44 Mb con sus respectivos archivos de estilo.

Para alimentar las bases de datos de las dos máquinas, se creó un programa que se encargó de insertar las 69 capas repetitivamente hasta que se ocupe el total de almacenamiento de cada máquina virtual. Como resultado, se ingresaron 2243 capas en cada máquina. Este programa generaba un nombre diferente de capa en cada inserción, con el objetivo

de que no existan registros con metadatos duplicados. La razón por la que se realizaron varias inserciones fue para analizar el comportamiento del componente de búsqueda con la mayor cantidad de capas posible en cada máquina.

4.2.3. Definición de escenarios de prueba

Previo a la ejecución de las pruebas, se definieron los escenarios a probar en cada módulo descrito en la introducción de este capítulo.

4.2.3.1. Búsquedas

Para esta sección se definió un escenario que involucraba todos los filtros de búsqueda posibles como texto, rangos de fecha, geolocalización, categorías y tipo de documento. El objetivo de este escenario es evaluar los resultados de los tiempos de búsqueda en ambas versiones de Data City.

Tabla 4.2 Escenario 1 de prueba para componente de búsquedas. Fuente: elaboración propia.

	Parámetros de búsqueda
Escenario 1	Texto
	Geolocalización
	Categoría
	Fecha
	Tipo de documento

4.2.3.2. Graficación de mapas

Para esta sección se definieron escenarios que nos permitieron evaluar el comportamiento del componente de graficación tanto para capas ráster como para capas vectoriales.

Con el objetivo de evaluar el comportamiento del componente de graficación se eligieron las capas de mayor tamaño del conjunto de capas disponible, tanto para capas ráster como para capas vectoriales, cada una con el correspondiente estilo.

Tabla 4.3 Escenario 2 de prueba para componente de graficación de mapas. Fuente: elaboración propia.

	Tipo de capa	Tamaño
Escenario 2	Ráster	44.6 MB.
		44.5 MB.
		44 MB

Tabla 4.4 Escenario 3 de prueba para componente de graficación de mapas. Fuente: elaboración propia.

	Tipo de capa	Tamaño
Escenario 3	Vectorial	1.3 MB.
		1 MB.
		0.7 MB

4.2.4. Ejecución de escenarios de prueba

Esta sección incluyó la ejecución de cada uno de los escenarios planteados en la sección 4.2.3.

4.2.4.1. Búsquedas

Para la ejecución de pruebas en el componente de búsqueda, se utilizó el inspector de trafico de red del navegador Google Chrome para obtener el tiempo de carga total desde que se envió la solicitud para realizar la búsqueda hasta que se listaron todos los resultados. En la Tabla 4.5 se puede visualizar los resultados

de la ejecución del escenario planteado en la sección 4.2.3. El tiempo total de carga mostrado a continuación fue el promedio de tiempos que se obtuvo luego de realizar 25 ejecuciones del mismo escenario.

Tabla 4.5 Resultados de las pruebas realizadas sobre el componente de búsqueda. Fuente: elaboración propia.

	Data City v1		Data City v2		
	Promedio	Desviación Estándar	Promedio	Desviación Estándar	Porcentaje de diferencia de tiempos
Escenario 1	520 ms	3.44	420 ms	2.72	-19.23%

Se evidenció que los tiempos de búsqueda utilizando Data City v2 disminuyeron en un 19.23% en comparación a los tiempos en Data City v1. La razón por la que Data City v2 presenta mejores resultados al momento de realizar búsquedas, es debido a que el componente CKAN de la nueva arquitectura implementa Solr como motor de búsqueda, el cual está optimizado para realizar búsquedas sobre grandes cantidades de datos.

4.2.4.2. Graficación de mapas

Para la ejecución de pruebas sobre la sección de graficación de mapas se utilizó la aplicación web GTmetrix [23]. Esta herramienta permitió realizar automáticamente varias solicitudes, desde servidores en diferentes partes del mundo, a un servidor

web específico con el objetivo de evaluar parámetros como tiempo total de carga, tamaño total de la página, entre otras métricas importantes que obtuvimos en esta sección.

A través de GTmetrix se ejecutaron entre 168 y 176 solicitudes, desde 3 servidores ubicados en Australia, Canadá y Brasil, a URLs correspondientes de cada capa descrita en el escenario de la sección 4.2.3.2. A continuación, se detallan los resultados obtenidos en las pruebas de los escenarios 2 y 3.

Tabla 4.5 Resultados para graficación de capas ráster.

Fuente: elaboración propia.

Escenario 2				
Tamaño [MB]	**Cantidad de Solicitudes**	**Tiempo de graficación promedio**		**Porcentaje de diferencia de tiempos**
		Data City v1	**Data City v2**	
44.6	168	61.2 s.	2.7 s.	-95.59%
44.5	176	60.9 s.	2.4 s	-96.06%
44	170	60.6 s.	2.2 s.	-96.36%
Promedio Total		60.9 s.	2.4 s.	-96.05%
Desviación Estándar		0.24	0.21	

Los resultados de la tabla 4.5, revelan que el tiempo de graficación de capas ráster en Data City v2 presenta una mejora porcentual de 96.05% sobre Data City v1. En base a estos resultados se puede afirmar que las modificaciones realizadas en el algoritmo del componente Tile Map Server sí redujeron los tiempos de graficación para capas ráster.

Tabla 4.6 Resultados para graficación de capas vectoriales.

Fuente: elaboración propia.

Escenario 3				
Tamaño [MB]	**Cantidad de Solicitudes**	**Tiempo de graficación promedio**		**Porcentaje de diferencia de tiempos**
		Data City v1	**Data City v2**	
1.3	166	3.2 s.	3.3 s.	3.25%
1	170	3.1 s.	3.2 s	3.22%
0.7	172	2.9 s.	3 s.	3.44%
Promedio Total		3.1 s.	3.2 s.	3.22%
Desviación Estándar		0.1	0.1	

Mientras que en la tabla 4.6 se puede evidenciar que no existe una diferencia significativa entre los tiempos de graficación para capas vectoriales de Data City v1 y Data City v2. El incremento de 3.22% con la nueva arquitectura se debe a que ahora las capas vectoriales son solicitadas al componente CKAN a través de una solicitud http y esto incremente la latencia en el tiempo de respuesta.

CAPÍTULO 5

5. Conclusiones, recomendaciones y trabajos futuros

5.1. Conclusiones

- La modularización de Data City dio como resultado una aplicación basada en componentes. Este nuevo enfoque permitió la creación de componentes como Tile Map Server y Time Series Server.
- Las modificaciones realizadas sobre el componente Tile Map Server, permitieron la reducción de aproximadamente 10 veces el tiempo de graficación de mapas con capas ráster y capas vectoriales.
- La modularización del componente Time Series Server permite a cualquier cliente externo, asociado a Data City, obtener e ingresar datos relacionados a sensores. Además, estos datos pueden ser visualizados como gráficas de series de tiempo desde Data City.
- La integración de nuevas fuentes de datos al proyecto Data City, se realizó exitosamente a través de la implementación de la nueva arquitectura planteada en este proyecto y la utilización de CKAN como portal de datos. Como resultado, Data City tiene la capacidad de integrar cualquier fuente de datos externa.
- Data City soporta la graficación de capas ráster y vectoriales provenientes de nuevas fuentes de datos externas.

5.2. Recomendaciones

- Todos los documentos provenientes de fuentes de datos externas, deben ser ingresados utilizando el API que provee CKAN.
- Cada nueva fuente de datos externa que se asocie a Data City, debe poseer una Organización dentro de CKAN para guardar sus datos.
- Las capas ráster o vectoriales que provengan de fuentes de datos externas y deseen ser graficadas en Data City, deben poseer la meta data descrita en la Tabla 2.3.

- Utilizar al menos dos máquinas virtuales con características iguales o superiores a las descritas en la Tabla 3.1, para alojar toda la nueva arquitectura del proyecto Data City. Se recomienda que en la primera máquina virtual se aloje la aplicación web Data City y en la segunda se aloje CKAN, los componentes Tile Map Server, Time Series Server y todas las configuraciones de cacheo y balanceo de carga.
- Para obtener los resultados descritos en la Tablas 3.13, 3.14 y 3.15, se recomienda proveer a Data City de al menos 3 instancias del componente Tile Map Server en una máquina virtual con características iguales o superiores a las descritas en la Tabla 3.1.

5.3. Trabajos futuros

- Desarrollar una extensión de CKAN que permita convertir los datos de series de tiempo en documentos.
- Convertir el componente de graficación de capas ráster en una extensión de CKAN.
- Conectar todos los procesos de administración y visualización de series de tiempo con la API de CKAN.
- Implementar flujos de integración y entrega continua para disponer siempre de una versión actualizada del sistema Data City.
- Implementar una sección para registro de usuarios.
- Agregar un campo que permita ingresar datos geoespaciales en la importación de documentos.
- Mejorar la experiencia del usuario, aplicando principios de diseño, basados en interacción humano computador.

BIBLIOGRAFÍA

[1] «Resiliencia Climática para Durán, una iniciativa para el establecimiento de Ciudades Resilientes en Ecuador,» Instituto de Investigación Geológico y Energético, [En línea]. Available: http://www.geoenergia.gob.ec/resiliencia-climatica-para-duran-una-iniciativa-para-el-establecimiento-de-ciudades-resilientes-en-ecuador/. [Último acceso: 08 30 2019].

[2] «Municipio del Distrito Metropolitano de Quito,» [En línea]. Available: www.quito.gob.ec. [Último acceso: 30 08 2019].

[3] «Gestión del conocimiento,» [En línea]. Available: https://www.funcionpublica.gov.co/eva/conocimiento/500/501-explicacion.html. [Último acceso: 30 08 2019].

[4] Manoj Kumar, N., Goel, S. and Mallick, P., «Smart Cities in India: Features, Policies, Current Status, and Challenges,» *IEEE International Conference on Technologies for Smart-City Energy Security and Power (ICSESP-2018),* p. 4, 2019.

[5] Santos, P., Lourenco, T., Perez-Peniche, C., Calcada, T. and Aguiar, A., «UrbanSense: an Urban-scale Sensing Platform for the Internet of Things,» *IEEE International Smart Cities Conference (ISC2),* p. 6, 2016.

[6] Dias, P., Rodrigues, J., Aguiar, A. and David, G., «Planning and managing data for Smart Cities: an application profile for the UrbanSense project,» *IEEE International Smart Cities Conference (ISC2),* p. 2, 2018.

[7] Amorim, R., Castro, J., Rocha da Silva, J. and Ribeiro, C., «A comparison of research data management platforms: architecture, flexible metadata and interoperability. Universal Access in the Information Society, 16(4),» p. 851.

[8] «CKAN,» [En línea]. Available: https://ckan.org. [Último acceso: 30 08 2019].

[9] «Data Gov,» [En línea]. Available: https://data.gov.au. [Último acceso: 03 06 2019].

[10] «Project Open Data Metadata Schema,» [En línea]. Available: https://project-open-data.cio.gov/v1.1/schema/ . [Último acceso: 05 06 2019].

[11] «FileStore and file uploads,» CKAN, [En línea]. Available: https://docs.ckan.org/en/2.8/maintaining/filestore.html. [Último acceso: 26 08 2019].

[12] "Digital Guide," [Online]. Available: https://www.ionos.es/digitalguide/servidores/configuracion/apache-solr/. [Accessed 04 08 2019].

[13] « Apache Solr,» [En línea]. Available: https://lucene.apache.org/solr/. [Último acceso: 26 08 2019].

[14] «CKAN code architecture,» [En línea]. Available: https://docs.ckan.org/en/2.8/contributing/architecture.html. [Último acceso: 29 08 2019].

[15] "NGINX," [Online]. Available: https://www.nginx.com/. [Accessed 26 08 2019].

[16] «API guide,» CKAN, [En línea]. Available: https://docs.ckan.org/en/ckan-2.7.3/api/#ckan.logic.action.get.package_search . [Último acceso: 23 07 2019].

[17] «Installing CKAN from source,» CKAN, [En línea]. Available: https://docs.ckan.org/en/2.8/maintaining/installing/install-from-source.html. [Último acceso: 26 08 2019].

[18] «ckanext-spatial,» Geo related plugins for CKAN, [En línea]. Available: https://docs.ckan.org/projects/ckanext-spatial/en/latest/. [Último acceso: 26 08 2019].

[19] «ckanext-spatial,» [En línea]. Available: https://docs.ckan.org/projects/ckanext-spatial/en/latest/install.html. [Último acceso: 30 08 2019].

[20] «PostgreSQL,» PostgreSQL, [En línea]. Available: https://www.postgresql.org/. [Último acceso: 29 08 2019].

[21] «Organizations and authorization,» CKAN, [En línea]. Available: https://docs.ckan.org/en/ckan-2.7.3/maintaining/authorization.html. [Último acceso: 26 08 2019].

[22] "City Health Map," [Online]. Available: https://www.cityhealthmap.com/. [Accessed 26 08 2019].

[23] "GTMetrix," [Online]. Available: https://gtmetrix.com/. [Accessed 04 08 2019].

[24] «Enterprise Cloud Computing SaaS,» Oracle Cloud, [En línea]. Available: https://cloud.oracle.com/home. [Último acceso: 26 08 2019].