Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2025

DESARROLLO DE UNA API RESTFUL CON PERSISTENCIA XML PARA LA GESTIÓN Y FACTURACIÓN DE SERVICIOS DE INFRAESTRUCTURA EN LA NUBE PARA LA EMPRESA TECNOLOGÍAS CHAPINAS, S.A.

202200147 – Mario Rodrigo Balam Churunel

Resumen

El presente proyecto aborda el diseño y desarrollo de una solución de software integral para la empresa Tecnologías Chapinas, S.A., orientada a la gestión y facturación automatizada de servicios de infraestructura en la nube. La arquitectura se basa en un modelo cliente-servidor desacoplado, implementando un backend como una API RESTful con el microframework Flask y un frontend simulador con Django, ambos desarrollados en Python. Una particularidad clave del sistema es el uso de archivos XML como mecanismo de persistencia de datos, cumpliendo con un requisito académico que explora alternativas a las bases de datos tradicionales. El sistema procesa archivos XML de entrada para la configuración inicial de recursos, categorías y clientes, así como para el registro de consumos. Las funcionalidades principales incluyen la consulta de datos del sistema, la generación de facturas por períodos específicos y la creación de reportes en formato PDF para el análisis de ventas y el detalle de facturas. Como conclusión, el proyecto demuestra la viabilidad de integrar tecnologías modernas como Flask y Django para construir una aplicación full-stack funcional, validando el uso de XML para la persistencia en contextos de baja concurrencia y demostrando el dominio de protocolos HTTP y la manipulación de datos estructurados.

Palabras clave
API REST, Flask, Django, Persistencia XML, Facturación.

Abstract

This project addresses the design and development of a comprehensive software solution for Tecnologías Chapinas, S.A., focused on the automated management and billing of cloud infrastructure services. The architecture is based on a decoupled client-server model, implementing a backend as a RESTful API with the Flask microframework and a simulator frontend with Django, both developed in Python. A key particularity of the system is the use of XML files as the data persistence mechanism, fulfilling an academic requirement that explores alternatives to traditional databases. The system processes input XML files for the initial configuration of resources, categories, and clients, as well as for logging consumption. Core functionalities include querying system data, generating invoices for specific periods, and creating PDF reports for sales analysis and invoice details. In conclusion, the project demonstrates the feasibility of integrating modern

technologies like Flask and Django to build a functional full-stack application, validating the use of XML for persistence in low-concurrency contexts and showcasing proficiency in HTTP protocols and structured data manipulation. (Nota: Se recomienda una revisión profesional de esta traducción).

Keywords
REST API, Flask, Django, XML Persistence, Billing.

Introducción

En el contexto de la creciente digitalización, los servicios de infraestructura en la nube (IaaS) se han vuelto fundamentales para las empresas. La gestión y facturación precisa de estos servicios es un desafío técnico significativo. Este proyecto presenta el desarrollo de un sistema a medida para la empresa "Tecnologías Chapinas, S.A.", cuyo objetivo es automatizar el proceso de facturación basado en el consumo de recursos de nube. La solución se articula a través de una arquitectura moderna que separa la lógica de negocio, expuesta mediante una API RESTful desarrollada en Flask, de la interfaz de usuario, un simulador web construido con Django. Este ensayo documenta el diseño arquitectónico, las tecnologías empleadas, las funcionalidades implementadas y las conclusiones obtenidas durante el ciclo de desarrollo, destacando el uso de archivos XML como capa de persistencia de datos.

Desarrollo del tema

El sistema se desarrolló siguiendo una estructura modular para garantizar la escalabilidad y mantenibilidad. A continuación, se describen los componentes principales.

a. Arquitectura de la solución

La aplicación se divide en dos componentes principales que se comunican a través del protocolo HTTP:

Servicio 2 (Backend): Una API RESTful desarrollada con Flask en Python. Es el cerebro del sistema, responsable de toda la lógica de negocio: procesar archivos XML, gestionar la persistencia de datos en un archivo data.xml, realizar los cálculos de facturación y generar los reportes en PDF.

Programa 1 (Frontend): Un simulador web desarrollado con Django. Actúa como cliente de la API, proporcionando una interfaz gráfica para que el usuario pueda interactuar con el backend, cargar archivos, solicitar facturas y descargar reportes.

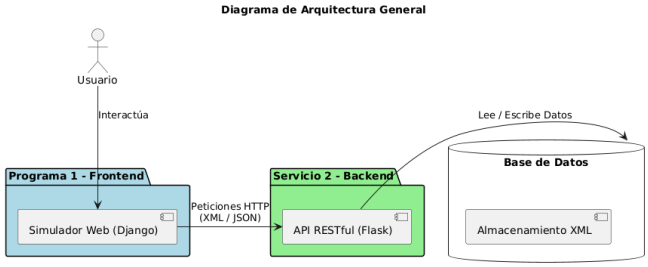


Figura 1. Arquitectura General de la Aplicación.

Fuente: elaboración propia.

b. Backend: API RESTful y Persistencia de Datos

El backend se construyó utilizando Flask por su ligereza y flexibilidad. La decisión más relevante, dictada por los requerimientos, fue utilizar un único archivo XML (data.xml) como base de datos. Para esto, se desarrolló un módulo de servicio (xml_manager.py) que contiene funciones para leer, escribir y modificar el árbol XML, utilizando la librería nativa de Python xml.etree.ElementTree.

Los principales endpoints implementados son:

- POST /api/cargarConfiguracion: Recibe un XML completo y sobrescribe la "base de datos".

- POST /api/registrarConsumo: Recibe un XML de consumos y actualiza las instancias correspondientes.
- GET /api/consultarDatos: Lee data.xml y lo devuelve como JSON.
- POST /api/generarFactura: Realiza los cálculos de facturación para un rango de fechas.
- POST /api/reporteVentas: Genera el PDF de análisis de ventas.

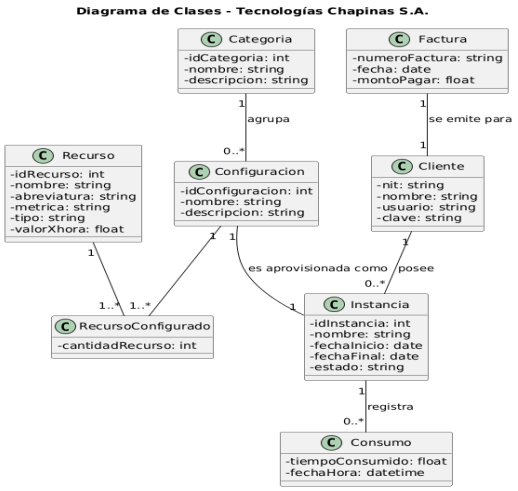


Figura 2. Diagrama de Clases del Sistema.

Fuente: elaboración propia.

c.Frontend: Simulador Web con Django

El frontend se desarrolló con Django, siguiendo el patrón Modelo-Vista-Template (MVT). Su función es simular la interacción de un usuario final con el sistema. Las vistas (views.py) se encargan de recibir las peticiones del usuario, comunicarse con la API del backend utilizando la librería requests, y pasar los datos de respuesta a las plantillas (templates).

Las funcionalidades implementadas incluyen:

- Formularios para la carga de archivos XML de configuración y consumo.
- Una página de consulta que presenta en tablas los datos actuales del sistema.
- Un formulario para solicitar la generación de facturas y reportes por rango de fechas.
- Un botón para resetear el sistema, eliminando el data.xml del backend.

d. Lógica de Facturación y Reportes

El núcleo de la lógica de negocio reside en la función generar_facturacion_detallada. Esta función recorre los consumos registrados para cada instancia de cada cliente. Utilizando expresiones regulares (re.search) para extraer las fechas de los consumos, filtra aquellos que caen en el rango solicitado. Posteriormente, cruza la información de la instancia con su configuración y la tabla de precios de recursos para calcular el costo de cada consumo.

Finalmente, agrega los costos para generar las facturas totales. Para la generación de reportes en PDF se utilizó la librería ReportLab, que permite construir documentos complejos con tablas y estilos a partir de los datos procesados.

Conclusiones

El desarrollo de este proyecto permitió validar la eficacia de una arquitectura de microservicios con una API RESTful como nexo entre el backend y el frontend. La elección de Flask y Django demostró ser adecuada, permitiendo un desarrollo rápido y modular.

El uso de XML como mecanismo de persistencia, aunque funcional para los propósitos académicos del proyecto y para volúmenes de datos pequeños, evidencia limitaciones significativas en un entorno de

producción real, principalmente en cuanto a eficiencia en las búsquedas y manejo de concurrencia. Una implementación futura debería migrar esta capa a un sistema de gestión de bases de datos relacional (como PostgreSQL) o NoSQL.

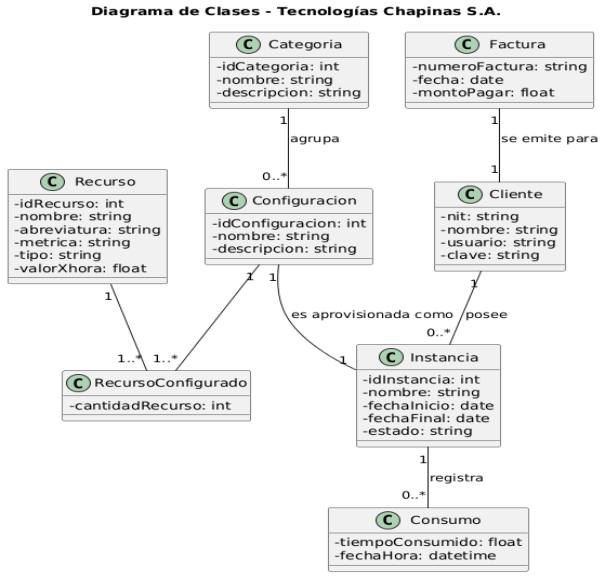
Se concluye que el sistema cumple con todos los requisitos funcionales solicitados, proporcionando una solución completa para la carga, consulta, facturación y reporte de los servicios de "Tecnologías Chapinas, S.A.", y sirviendo como una demostración práctica de un ciclo de desarrollo de software full-stack.

Referencias bibliográficas

- Django Software Foundation. (2025). Django Documentation. Recuperado de https://docs.djangoproject.com/
- Flask Documentation. (2025). Flask Project. Recuperado de https://flask.palletsprojects.com/
- Grinberg, M. (2018). Flask Web Development, 2nd Edition. O'Reilly Media, Inc.
- Python Software Foundation. (2025). xml.etree.ElementTree — The ElementTree XML API. Recuperado de https://docs.python.org/3/library/xml.etree.elementtree.html
- ReportLab. (2025). ReportLab User Guide. Recuperado de https://www.reportlab.com/docs/reportlab-userguide.pdf

- **Anexos**

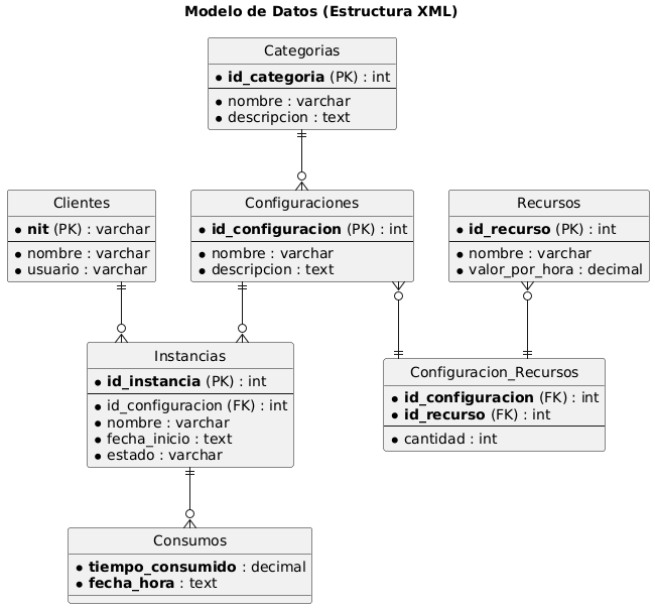
Diagrama de Clases del Sistema.



Anexo1. Diagrama de Clases del Sistema.

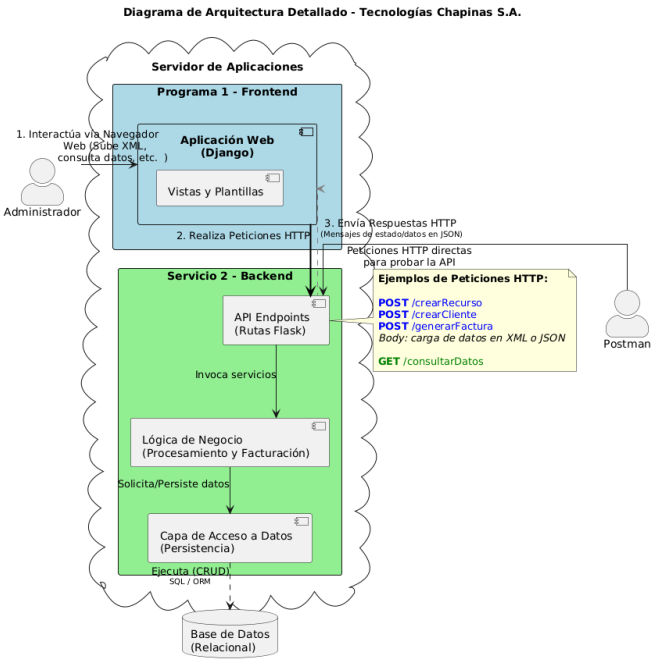
Fuente: elaboración propia.

Modelo de Datos (Representación Entidad-Relación del XML.



Anexo2. Diagrama de Modelo de datos (Estructura XMLm

Fuente: elaboración propia.



Anexo3. Diagrama de Arquitectura Detallado

Fuente: elaboración propia.