Bases de Datos I

Tarea Programada III y IV

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

IC-4301 | Bases de Datos I

Prof. Franco Quirós Ramírez

Estudiantes:

Elías Ramírez Hernández - 2024090300

Andrés Alejandro Rodríguez Brizuela - 2024256537

Andrés Baldi Mora - 2024088934

Miércoles 26 de noviembre

II Semestre 2025

Tabla de contenidos

Índice de figuras

Capítulo 1: Introducción

En este documento se presenta la documentación técnica y el análisis de resultados de la tercera tarea programada del curso de Bases de Datos I. El objetivo principal de este informe es describir el diseño, la implementación y el comportamiento de un sistema de facturación municipal orientado a la gestión de propiedades, conceptos de cobro y pago de facturas, siguiendo las especificaciones brindadas en el enunciado del proyecto.

El sistema modela el funcionamiento de una municipalidad en cuanto al cobro de servicios como impuesto a la propiedad, consumo de agua, recolección de basura, mantenimiento de parques, intereses moratorios y reconexiones de agua, entre otros. Las propiedades se asocian a personas (propietarios), se les asignan conceptos de cobro (CC) de forma automática mediante triggers, y se generan facturas mensuales según la fecha de inscripción de la propiedad.

Adicionalmente, el sistema procesa archivos XML que simulan el comportamiento diario de la municipalidad a lo largo de varios meses, incorporando:

- Lecturas y ajustes de medidores de agua.
- Pagos de facturas realizados a través de nodos XML.
- Cambios de valor fiscal de las propiedades.
- Asociación y desasociación de propietarios.

El proyecto incluye también un portal web administrativo, que permite a un usuario de tipo administrativo buscar propiedades por número de finca o por cédula de propietario, visualizar las facturas pendientes y gestionar el pago de la factura más antigua. Durante el proceso de pago se calculan, cuando corresponde, intereses moratorios y se actualiza el estado de las facturas.

Capítulo 2: Contexto del Desarrollo

2.1. Descripción del Ambiente de Desarrollo

Para la realización de este proyecto se utilizaron las mismas herramientas colaborativas y tecnologías de la segunda tarea programada, adaptadas ahora al contexto de la facturación municipal:

- **Control de Versiones**

  Se utilizó GitHub para gestionar el código fuente del proyecto, permitiendo trabajo colaborativo, ramas de desarrollo y revisión de cambios.
  El repositorio del proyecto se mantuvo de forma privada durante la realización del proyecto (link acá), en este momento, se encuentra público y organizado por carpetas para el procesamiento de XML, scripts SQL y frontend/backend.

- **Entorno de Desarrollo Integrado (IDE)**

  Se utilizó Visual Studio Code como editor principal para:
  - Scripts en Python.
  - Archivos HTML, CSS y JavaScript del frontend.

- **Gestión de Base de Datos**

  - Se utilizó SQL Server Management Studio (SSMS) para administrar la base de datos del proyecto.
  - El motor de Microsoft SQL Server se ejecuta en una instancia central, en nuestro caso, en una máquina anfitriona, de la cúal fue elegido el computador de Elias, a este se conectan los miembros del equipo.
  - Toda la lógica de acceso a datos y su manipulación, se implementó mediante procedimientos almacenados y triggers.

- **Red y Conectividad**

  - Se utilizó Hamachi para simular una red local virtual, facilitando que todos los integrantes pudieran conectarse al mismo servidor de base de datos y al backend, incluso trabajando desde redes físicas distintas.

- **Comunicación y Coordinación**
    - Las reuniones de coordinación del equipo se realizaron por Discord, donde se discutieron decisiones de diseño, división de tareas y revisiones de avance.
- **Bitácora de Trabajo**
    - Se utilizó Blogger como bitácora para registrar sesiones de trabajo, decisiones importantes y problemas encontrados, de forma similar a la segunda tarea. El link al Blogger es este.

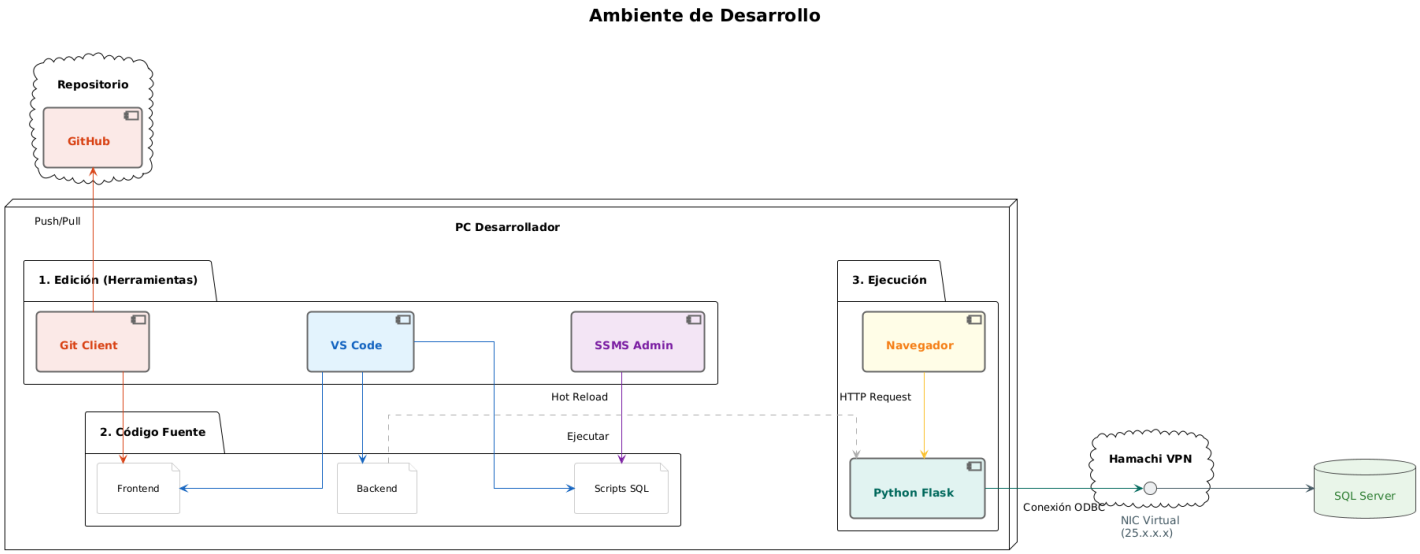

Figura 1: Diagrama de Ambiente de Desarrollo del Proyecto

2.2 Arquitectura de la Aplicación

El sistema se desarrolló nuevamente bajo una arquitectura de tres capas, adaptada al dominio de facturación municipal.

Capa de Presentación (Frontend)

La capa de presentación consiste en una aplicación web orientada al usuario administrativo de la municipalidad. Sus responsabilidades principales son:

- Permitir que el usuario administrativo inicie sesión (en caso de que se requiera autenticación explícita).

- Buscar propiedades:
    - Por número de finca.
    - Por número de identificación del propietario.
- Mostrar los datos básicos de la propiedad seleccionada como el número de finca, zona, uso y valor fiscal.
- Listar las facturas pendientes asociadas a la propiedad, ordenadas desde la más antigua a la más reciente.
- Permitir seleccionar la factura pendiente más antigua y lanzar el proceso de cálculo y confirmación del pago:
    - Mostrar monto base.
    - Mostrar intereses moratorios calculados al día de operación (cuando aplica).
    - Mostrar el total a pagar antes de la confirmación.

Esta capa se implementó con:

- HTML para la estructura de las páginas.
- CSS para los estilos y diseño de la interfaz.
- JavaScript para manejar la lógica de interacción.

Capa de Lógica (Backend)

La capa de lógica de negocio se implementó en Python, utilizando el microframework Flask para exponer una API REST. Esta capa tiene dos grandes responsabilidades:

1. **Procesamiento de XML**
    - Carga del archivo de catálogos (catalogosV2.xml), para inicializar en la base de datos:
        - Conceptos de cobro (CC) y sus parámetros.
        - Tipos de uso y zona de propiedad.
        - Tipos de movimiento, medio de pago, periodo y tipo de monto de los CC.

        - Parámetros globales como días de vencimiento de factura y días de gracia para la corta.

    - Procesamiento del archivo de operaciones (xmlUltimo.xml):

        - Alta de personas y propiedades.
        - Asociación/desasociación de propietarios con propiedades.
        - Asignación manual de algunos CC cuando el XML lo especifica.
        - Lecturas y ajustes de medidores de agua.
        - Pagos de facturas provenientes del XML.
        - Cambios de valor fiscal de las propiedades.

    - Por cada <FechaOperacion> del XML, el backend llama a procedimientos almacenados que:

        - Ejecutan procesos masivos de generación de facturas.
        - Generan órdenes de corta y reconexión de agua.
        - Actualizan bitácoras y saldos de medidores.

2. **Portal Administrativo**

    - Exposición de endpoints REST para:

        - Buscar propiedades por número de finca o por cédula de propietario.
        - Recuperar la lista de facturas pendientes para una propiedad determinada.
        - Calcular el monto total a pagar de la factura más antigua (incluyendo intereses moratorios si la factura está vencida).
        - Confirmar el pago, registrando el medio de pago y actualizando el estado de la factura.

    - Manejo de errores:

        - Captura de excepciones y retorno de códigos de error.
        - Consulta de mensajes descriptivos en la tabla de errores (si aplica una estructura similar a la usada en el proyecto anterior).

La comunicación con la base de datos se realiza exclusivamente a través de pyODBC, enviando parámetros a los SPs y recibiendo resultados tabulares o códigos de salida.

Capa de Datos (Base de Datos)

La capa de datos se implementó sobre Microsoft SQL Server. Entre sus características destacadas se encuentran:

- Modelo relacional que contempla tablas para:
    - Personas, usuarios y propietarios.
    - Propiedades, tipos de uso y tipo de zona.
    - Medidores de agua y movimientos de lectura/ajuste.
    - Conceptos de cobro (CC) y sus parámetros.
    - Asociación propiedad-concepto de cobro.
    - Facturas y detalle de facturas.
    - Órdenes de corta y reconexión.
    - Pagos y comprobantes de pago.
    - Parámetros globales del sistema.
    - Tablas de bitácora para registrar cambios relevantes.
- Uso intensivo de procedimientos almacenados para:
    - Cargar catálogos desde XML.
    - Procesar altas y movimientos del XML de operaciones.
    - Generar facturas diarias según la fecha de registro de cada propiedad.
    - Calcular el consumo de agua a partir de los saldos de m^3.
    - Aplicar intereses moratorios cuando proceda.
    - Generar y actualizar órdenes de corta y reconexión.
    - Gestionar los pagos provenientes tanto del XML como del portal administrativo.

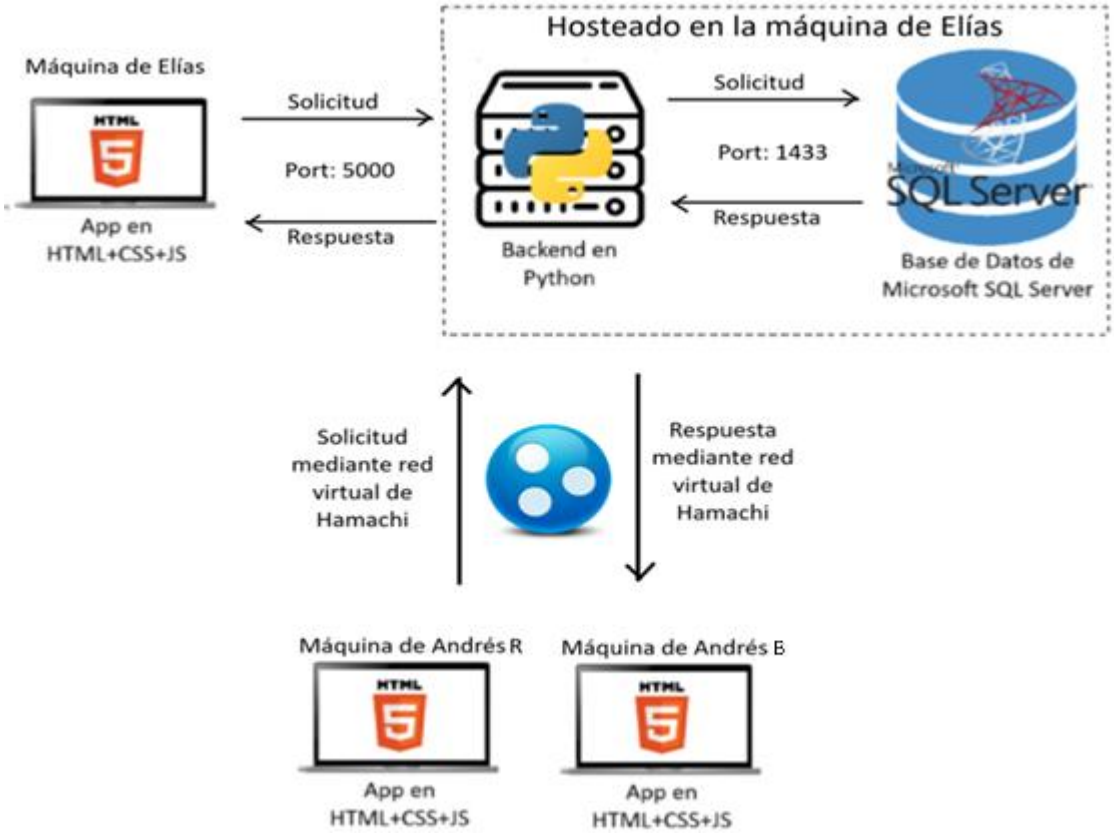

Figura 2: Arquitectura general del sistema de facturación municipal.

2.3 Tecnologías Empleadas y Explicación

Frontend

- **HTML**

  Permite definir la estructura de las pantallas del portal administrativo, incluyendo formularios para búsqueda de propiedades, tablas para mostrar facturas pendientes y botones para disparar el proceso de pago.

- **CSS**

  Se utiliza para dar estilo y layout a la aplicación web. Permite separar la lógica y la presentación, logrando una interfaz más limpia y fácil de usar.

- **JavaScript**

  - Gestiona la lógica de interacción en el navegador.
  - Manipula la actualización de los datos mostrados (lista de facturas, totales calculados, mensajes de confirmación o error).

Backend

- **Python**
  Lenguaje principal para implementar la lógica de negocio del proyecto, tanto para el procesamiento de los archivos XML como para la API del portal administrativo.

- **Flask**
  - Microframework web utilizado para levantar la API REST.
  - Provee el enrutamiento de endpoints y el manejo básico de solicitudes HTTP.

- **Flask-CORS**
  - Permite habilitar el CORS para que el frontend.

- **pyODBC**
  - Librería para conectarse a SQL Server desde Python.
  - Permite llamar procedimientos almacenados, enviar parámetros, leer conjuntos de resultados y manejar códigos de retorno.

Base de Datos

- **Microsoft SQL Server**
  - Motor relacional usado para almacenar toda la información del sistema de facturación municipal.
  - Se aprovechan características como transacciones, triggers y procedimientos almacenados para garantizar integridad, atomicidad de procesos masivos y rendimiento.

- **T-SQL**
  - Lenguaje empleado para la definición de tablas, constraints, procedimientos almacenados y triggers.
  - Las reglas de negocio críticas como la facturación, cálculo de consumo, intereses, cortas y reconexiones, se centralizan en esta capa.

Capítulo 3: Análisis de Resultados del Proyecto

En este capítulo se presenta una tabla para evaluar el grado de cumplimiento de los requerimientos funcionales y no funcionales de la tercera tarea programada, siguiendo el estilo del proyecto anterior.

La descripción de cada elemento se basa en los requerimientos del sistema de facturación municipal: procesamiento de XML, generación de facturas, aplicación de reglas de negocio mediante triggers y SPs, y portal administrativo de pagos.

Elemento Evaluado	Valoración	Porcentaje de Implementación	Comentarios
Documentación y Diseño	Implementado exitosamente	100%	La documentación cumple todos los requisitos.
Base de Datos y Modelo Físico	Implementado exitosamente	100%	Se crearon todas las tablas según el modelo físico y se cargaron los datos desde el XML.
R1: Carga de catálogos desde XML	Implementado exitosamente	100%	Se cargan los catálogos correctamente.
R2: Procesamiento del XML de operaciones por fecha	Implementado exitosamente	100%	Se procesa correctamente cada operación mediante esta simulación.
R3: Triggers de asignación automática de CC	Implementado exitosamente	100%	Implementados correctamente.
R4: Generación mensual de facturas	Implementado exitosamente	100%	Se implementó el job diario correctamente.

R5: Cálculo de consumo de agua y cargos de ConsumoAgua	Implementado exitosamente	100%	Se implementó correctamente.
R6: Cálculo y aplicación de intereses moratorios	Implementado exitosamente	100%	Se implementó el job diario correctamente.
R7: Generación y gestión de órdenes de corta	Implementado exitosamente	100%	Se implementó el job diario correctamente.
R8: Generación y gestión de órdenes de reconexión	Implementado exitosamente	100%	Se implementó el job diario correctamente.
R9: Pagos desde XML	Implementado exitosamente	100%	Se procesa correctamente cada operación mediante esta simulación.
R10: Portal administrativo de pagos	Implementado exitosamente	100%	Implementada la interfaz de acuerdo con los requerimientos.
R11: Bitácoras y trazabilidad	Implementado exitosamente	100%	Implementada la tabla Bitacora correctamente.
R12: Manejo de errores y mensajes	Implementado exitosamente	100%	Tablas de Error, DBError incorporadas.
R13: Uso exclusivo de SPs y transacciones	Implementado exitosamente	100%	Se hace correcto manejo del todo o nada.

Capítulo 4: Métricas del Proyecto

En este capítulo se incluyen métricas cuantitativas del proyecto como horas, commits, líneas de código, etc.

Tabla de Métricas Globales

Métrica	Total	Comentarios
Horas trabajadas	32	Tiempo total de desarrollo y pruebas.
Número de sesiones de trabajo	13	Cantidad de sesiones de programación en pareja o individuales.
Duración total de las pruebas (en horas)	5	Tiempo dedicado a pruebas de integración y casos extremos.
Horas de resolución de problemas	2	Tiempo aproximado invertido en depurar errores complejos.
Total, de líneas de código	7352	Suma de SQL, Python, HTML, CSS y JavaScript.
Número de commits en GitHub	73	Métrica a obtener de las estadísticas del repositorio.
Cantidad de archivos	46	Archivos SQL, scripts Python, JS, HTML, CSS, configuraciones, etc.
Cantidad de datos de prueba	3452	Número de registros generados por el XML en tablas clave (facturas, pagos).
Cantidad de tablas creadas	32	Tablas del modelo (propiedades, facturas, CC, bitácoras, etc.).
Procedimientos almacenados	32	SPs para carga de XML, facturación, pagos, cortas, reconexiones, etc.

Triggers implementados	7	Triggers de asignación de CC y bitácoras.
Endpoints o funciones de backend	28	Endpoints REST del portal administrativo y/o funciones auxiliares en Python.
Cantidad de facturas generadas	310	Número total de facturas emitidas en la simulación.
Cantidad de órdenes de corta/reconexión	517	Total de órdenes generadas durante la simulación.

Métricas de GitHub

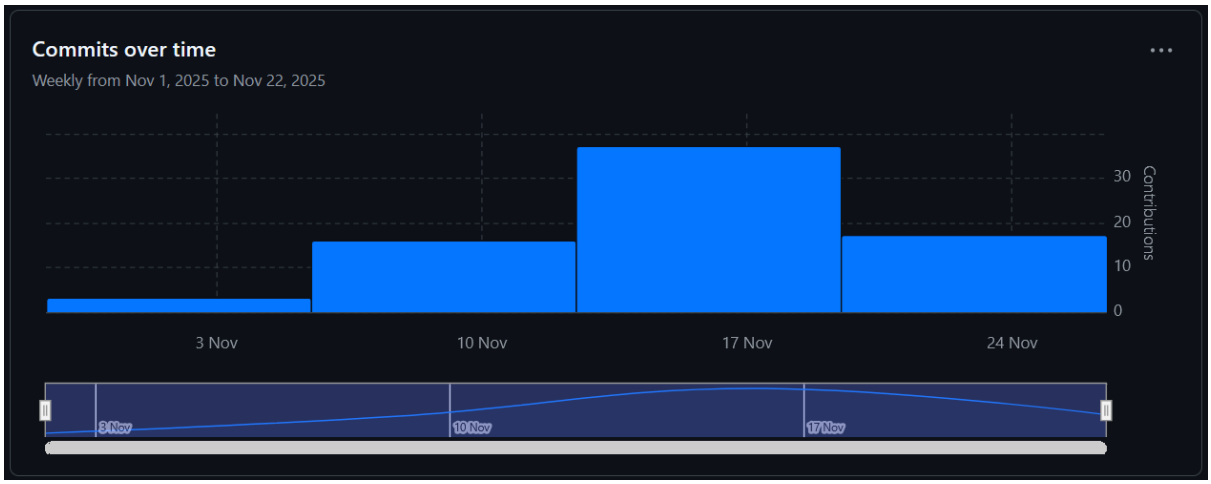

Figura 3: Gráfico de contribuciones del repositorio.

Figura 4: Gráfico de frecuencia de contribuciones del repositorio.

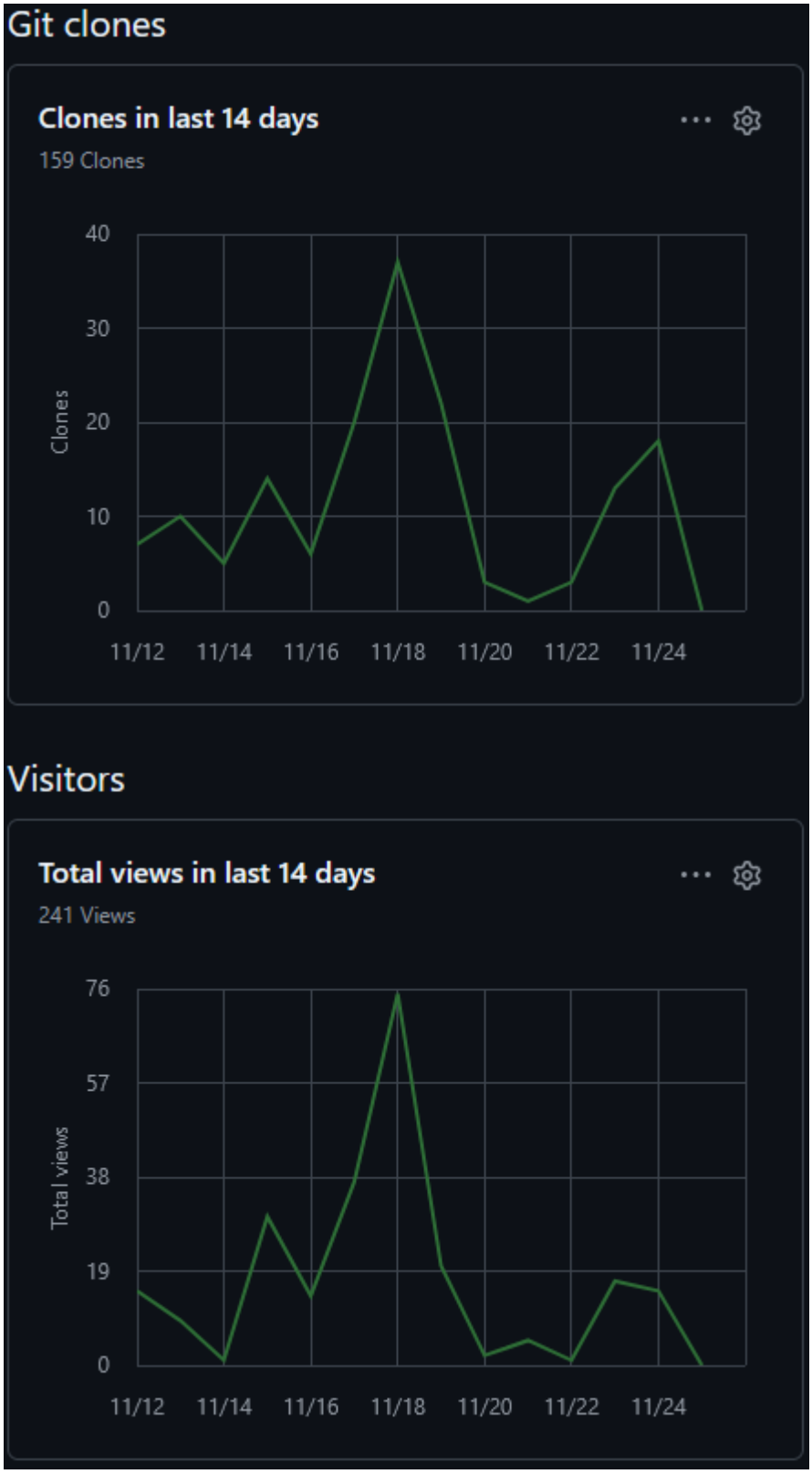

Figura 5: Gráfico de frecuencia de clonaciones y visitas del repositorio.