Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, Escuela de Vacaciones, Junio 2021.

Desarrollo de una REST API que se comunique con una aplicación en Django

201930566 - Alexander Manuel de Jesús Tzoc Alvarado

Resumen

La comunicación por medio de dos aplicaciones tiene diferentes formas de hacerlo. Hoy en día se utilizan diferentes protocolos, pero el más utilizado es HTTP, este cuenta con varios métodos de comunicación que hacen la vida del programador mucho más sencillas, los métodos utilizados en este proyecto son GET y POST, el primero para solicitar datos al servidor y el segundo para enviar datos al servidor. El proyecto está separado en dos partes, la primera es una REST API que es la encargada de la persistencia de datos en un archivo XML y no tiene una interfaz gráfica, también puede ser consultada por POSTMAN. La segunda parte del proyecto es una aplicación web que es la encargada de comunicarse la API, en la aplicación web se pueden consultar los datos de clientes, juegos, cargar nuevos datos y ver errores de la carga de datos. Los datos están en formato CSV y son transformados a formato XML.

Palabras clave
XML, HTTP, Consulta, Respuesta, Servidor

Abstract

Communication through two applications has different ways of doing it. Nowadays different protocols are used, but the most used is HTTP, it has several communication methods that make the life of the programmer much easier. The methods used in this project are GET and POST, the first one requests data from the server and the second one sends data to the server. The project is separated into two parts, the first one is a REST API that is responsible for the persistence of data in an XML file and does not have a graphical interface, it can also be consulted by POSTMAN. The second part of the project is a web application that is in charge of requesting and sending data from the API. In the web application it is possible to consult the data of clients, games, load new data and see data load errors. The data is in CSV format and is transformed into XML format.

Keywords
XML, HTTP, Request, Response, Server

Introducción

La realización de una aplicación web cuenta con varias partes, desde definir las vistas y el mejor diseño para que sea lo más sencillo de usar para el usuario, hasta la planeación del manejo de datos y los protocolos para transportar dichos datos. Es necesario tener en cuenta todos estos aspectos para que el proyecto pueda realizarse de manera óptima. También existen aplicaciones que simplemente se encargan de manejar datos y no tienen una interfaz gráfica, son las llamadas APIs (*Application Programming Interface*), estas son muy utilizadas para separar lógicamente los datos y la parte gráfica. La aplicación web se comunica con la API y esta le retorna los datos para que pueda mostrar. Los datos son cadenas de texto con un cierto formato acordado entre la API y la aplicación web. Existen varios formatos, los más utilizados son JSON y XML.

Desarrollo del tema

Se tiene como tarea desarrollar una solución integral que implemente una API que brinde servicios utilizando el protocolo HTTP bajo el concepto de programación orientada a objetos (POO) y la persistencia de datos. Comencemos definiendo el concepto de una API.

Según un artículo publicado en la página oficial de *Red Hat* "Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. API significa: interfaz de programación de aplicaciones" (*¿Qué es una API?*, s. f.). Las APIs son las encargadas de transportar datos. No cuentan como tal con una interfaz gráfica ya que no la necesita para su funcionamiento. Para el desarrollo de la API se utilizó el *framework* (marco de trabajo) ***FLASK***. Este framework facilita mucho el trabajo ya que cuenta con varias funcionalidades que ahorran mucho trabajo a la hora de desarrollar una API. Esta API cuenta diferentes funcionalidades dependiendo de la ruta que sea consultada.

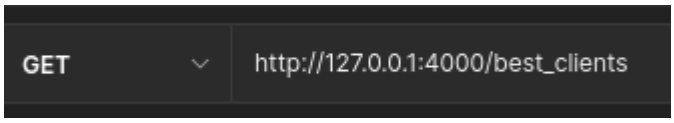

Figura 1. Consulta de api

Fuente: Elaboración propia

En la figura 1 se puede observar una de las rutas, esta ruta es la encargada de retornar los mejores clientes ordenamos de manera descendente por la cantidad gastada en la tienda *CHET.* Existen diferentes rutas para consultar: */most_selled (juegos más vendidos), /games (retorna todos los juegos en el inventario de la tienda), /birthdays (retorna todos los clientes de los cuales se tenga registro de su fecha de cumpleaños)*. Todas las rutas mencionadas son accedidas por el método GET:

> *"El método GET envía la información codificada del usuario en el header del HTTP request, directamente en la URL. La página web y la información codificada se separan por un interrogante" (Lázaro, 2018)*

En este caso no el único dato enviado en la url fue la ruta base de la consulta. Todas las consultas retornaban una cadena de texto con formato XML, este formato es muy flexible y fácilmente manipulable. XML consiste en un lenguaje de marcado creado por el **W3C** (World Wide Web Consortium), con la finalidad de definir una sintaxis para la codificación de documentos, que tanto los usuarios como las propias máquinas en sí puedan ser capaces de leer (colaboradores de Wikipedia, 2021).

```
<mejoresClientes>
    <cliente>
        <nombre>Jose</nombre>
        <edad>-</edad>
        <fechaCumpleaños>-</fechaCumpleaños>
        <fechaPrimeraCompra>-</fechaPrimeraCompra>
        <fechaUltimaCompra>10/06/2021</fechaUltimaCompra>
        <cantidadComprada>6</cantidadComprada>
        <cantidadGastada>10000</cantidadGastada>
    </cliente>
    <cliente>
        <nombre>Jose</nombre>
        <edad>-</edad>
        <fechaCumpleaños>-</fechaCumpleaños>
        <fechaPrimeraCompra>-</fechaPrimeraCompra>
        <fechaUltimaCompra>10/06/2021</fechaUltimaCompra>
        <cantidadComprada>6</cantidadComprada>
        <cantidadGastada>10000</cantidadGastada>
    </cliente>
```


Figura 2. Formato del response

Fuente: Elaboración propia

Para el ingreso de nuevos datos se manda por método POST, que es un método más seguro que el GET y generalmente se usa para la comunicación entre el Frontend (interfaz de usuario) y el Backend (Parte lógica de la aplicación), en esta consulta se envía una cadena de texto con formato XML y *tags* espaciales. Si los datos son correctamente enviados, los datos serán guardados con otro formato que permitirá responder a las consultas de una manera más óptima.

Para la segunda parte del proyecto, la aplicación web, se utilizó el framework **Django** que es un framework más completo y permite realizar aplicaciones tan complejas como sencillas. En este caso **Django** utiliza un sistema de vistas y templates que son los encargados de mostrar los datos necesarios al usuario. Para los templates se utilizó el lenguaje HTML. Según la página oficial de *CódigoFacilito* "HTML es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de la siglas que corresponden a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto. Muy parecido a XML, pero con etiquetas previamente definidas por el lenguaje". (Flores Herrera, 2015)

Para una mejor visualización de los datos se utilizó CSS. Hojas de Estilo en Cascada (del inglés Cascading Style Sheets) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML (en-US) (incluyendo varios languages basados en XML como SVG, MathML o XHTML). CSS describe cómo debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios." (CSS | MDN, 2021).

Además se necesitaba mostrar en gráficas los datos de los mejores clientes, juegos más vendidos y cantidad de juegos por categoría. Para una mejor visualización de las gráficas se utilizó la librería de *JavaScript Chart.js*, esta librería facilita mucho el trabajo de la visualización de los datos. En su documentación se encontró lo que se necesitaba para este proyecto. Para hacer funcionar esta librería, se hacía una petición a la API desde **Django** y luego por medio de los templates se pasaban los datos al *frontend*.

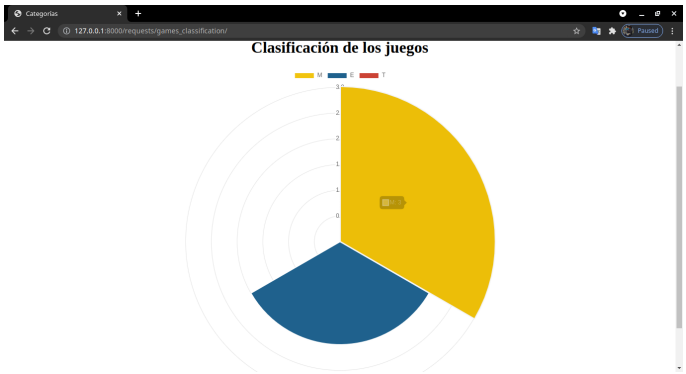

Figura 3. Gráfica de clasificación de juegos

Fuente: Elaboración propia

Antes de tener todos los datos cargados para su visualización se necesitan cargar los datos en

formato CSV. El formato CSV son valores separados por comas (Comma Separated Values por sus siglas en inglés), cada fila de un archivo csv representa una tupla y cada coma (y a veces punto y coma) representa una columna. Para leer estos archivos (4 en total) se validaron las cabeceras, si los archivos traen las cabeceras incorrectas entonces los archivos no serán leídos. Una vez validadas las cabeceras se procede a validar todos los campos utilizando expresiones regulares. Según la documentación oficial de Python "Una expresión regular (o RE, por sus siglas en inglés) especifica un conjunto de cadenas que coinciden con ella; las funciones de este módulo permite comprobar si una determinada cadena coincide con una expresión regular dada (o si una expresión regular dada coincide con una determinada cadena, que se reduce a lo mismo)." (*re — Operaciones con expresiones regulares — documentación de Python - 3.9.6*, s. f.). Se utilizaron diferentes expresiones regulares para la validación de diferentes campos.

```
PATTERNS = {
    'nombre': '[áéíóúa-zÁÉÍÓÚA-Z]{2,}\s?[áéíóúa-zÁÉÍÓÚA-Z]{1,}',
    'edad': '\d\d',
    'fecha': '\d{1,2}\/\d{2,2}\/\d{4,4}',
    'cantidad': '\d+',
    'gastado': '[0-9]+(\.[0-9]{1,2})?',
    'año': '\d{4,4}',
    'clasificacion': '[ETM]',
    'no_validate': '.+'
}
```

Figura 4. Expresiones regulares usadas en la aplicación web.

Fuente: Elaboración propia

Al momento de validar todos los campos, si existiese uno solo erróneo, no se procede con la transformación a formato XML. Casi todos los campos tienen que ser validados para evitar errores al momento de guardar y consultar los datos. Si un archivo contiene un error y ese error es encontrado al momento de validar los datos, entonces en el apartado de "Reporte de errores" aparecerá el tipo de error y los datos necesarios para que el usuario pueda corregir los datos.

Figura 5. Reporte de errores

Fuente: Elaboración propia

Si todos los datos son correctos, entonces se procede a transformar los datos a formato XML y mostrarle al usuario en un editor de texto, que para su elaboración se usó la librería de *JavaScript CodeMirror*, en este editor de texto el usuario puede editar, agregar y eliminar etiquetas. Si se hace una mala modificación, entonces los datos no se guardarán y el usuario será notificado. Si los datos están correctos el usuario cuenta con un botón para enviar la información y los datos guardados ya estarán disponibles para ser mostrados en las diferentes gráficas.

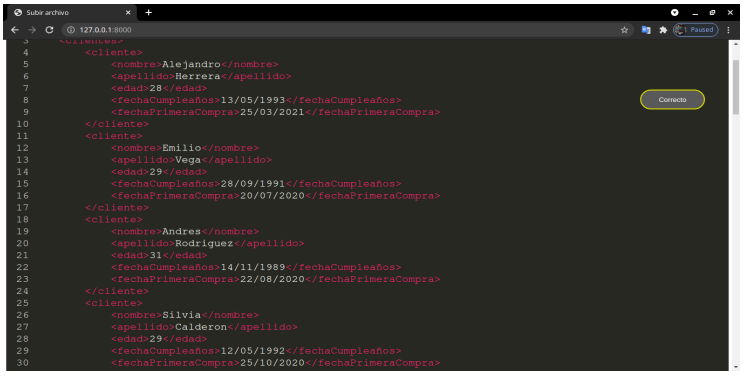

Figura 6. Editor de texto

Fuente: Elaboración propia

Cabe resaltar que para el correcto funcionamiento de la aplicación web, la API debe estar corriendo correctamente, de no ser así la aplicación web estará limitada a mostrar información como esta documentación y el nombre del programador (autor de esta documentación).

Conclusiones

La realización de una aplicación puede ser tan complicada como el programador o los programadores quieran hacerla. Obviamente depende de la complejidad de los problemas a resolver, pero sin una correcta planeación se puede volver un problema muy grande. Por eso es importante establecer el formato en el que las partes de la aplicación se comunican, el formato utilizado, la manera de guardar los datos, etc. JSON es un formato muy flexible y python brinda nativa|mente de las herramientas para poder manejarlo. XML es un formato muy sólido para manejar cantidades medianamente grandes de datos.

El uso de los protocolos HTTP facilita en gran medida la comunicación entre aplicaciones, no es un secreto que sea de los protocolos más usados en el mundo.

Aunque los datos estén bien lógicamente, también tienen que ser presentados de manera correcta al usuario para una mejor interpretación y comodidad de quien usa la aplicación. El Frontend también es importante.

Referencias bibliográficas

CSS | MDN. (2021, 19 junio). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/CSS

Flores Herrera, J. (2015, 25 agosto). Qué es HTML. CódigoFacilito. https://codigofacilito.com/articulos/que-es-html

Lázaro, D. (2018). GET y POST en PHP. Diego Lázaro. https://diego.com.es/get-y-post-en-php

re — Operaciones con expresiones regulares — documentación de Python - 3.9.6. (s. f.). docs.python. Recuperado 2 de julio de 2021, de https://docs.python.org/es/3/library/re.html

Apéndice

```
▼<Chet>
  ▶<clientes>
    ...
    </clientes>
  ▶<juegos>
    ...
    </juegos>
</Chet>
```

Figura 7. Formato final xml
Fuente: Elaboración propia

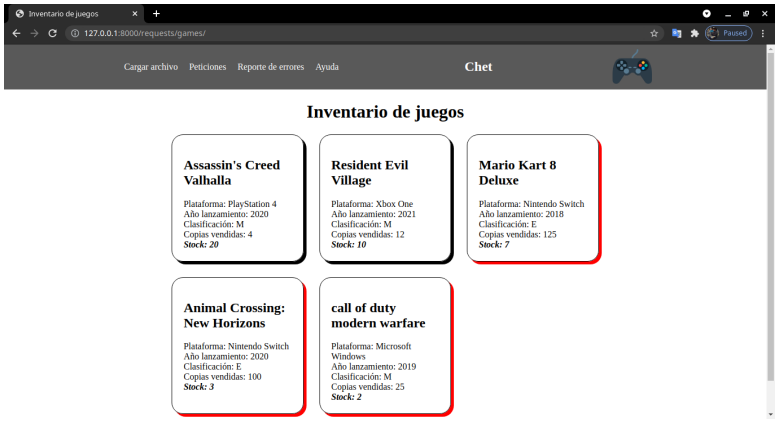

Figura 7. Listado de juegos
Fuente: Elaboración propia

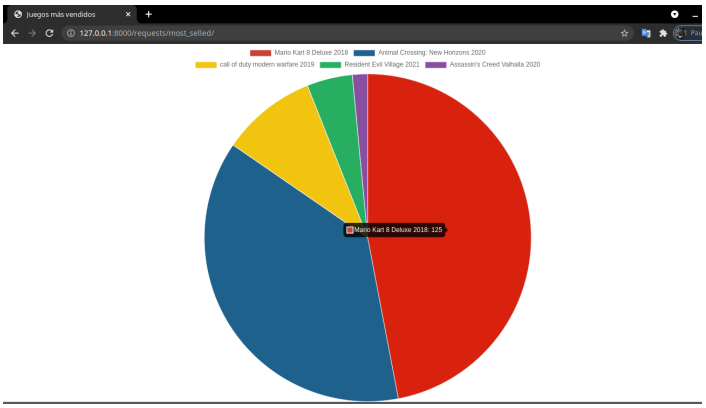

Figura 7. Gráfica de juego más vendidos
Fuente: Elaboración propia

```xml
<juego>
  <nombre>Mario Kart 8 Deluxe</nombre>
  <plataforma>Nintendo Switch</plataforma>
  <añoLanzamiento>2018</añoLanzamiento>
  <clasificacion>E</clasificacion>
  <fechaUltimaCompra>30/06/2021</fechaUltimaCompra>
  <copiasVendidas>125</copiasVendidas>
  <stock>7</stock>
  <color>#F00</color>
</juego>
```

Figura 7. Formato xml de juegos
Fuente: Elaboración propia