Segundo parcial fundamentos de informatica

Introduccion a Web:

**Internet:**

Internet es la red de redes de computadoras, conectadas por un cableado físico que permite el cambio de información entre los usuarios.

La computadora que proporciona el recurso o información se la llama servidor, a la que recibe se la llama cliente.

A través de internet se pueden enviar msj, programas ejecutables, archivos de texto, consultar base de datos, etc.

**La WEB:**

La web es un conjunto de paginas interconectadas por la que podemos navegar.

Las páginas están pensadas para consumir contenido hipertextual, que tiene enlaces a otros contenidos.

**Introducción al concepto de Cloud Computing:**

La computación en la nube es el consumo o prestación bajo demande de recursos tecnológicos a través de internet. Esto sirve para no tener que tener unos servidores en tu casa con centro de datos físicos, podes consumir estos servicios tecnológicos, como potencia informática, almacenamiento y bases de datos, según te sea necesario, de un proveedor.

**Arquitectura Web:**

Los sistemas web de hoy en día tienen una arquitectura distribuida, sus componentes están separados en dos tipos de nodos distintos: clientes (muchos) y servidores (uno). En esta arquitectura, llamada cliente-servidor, los clientes se comunican con el servidor siguiendo un protocolo de pedido-respuesta en el que: 1. El cliente hace un pedido 2. El servidor lo procesa y responde 3. El cliente lo presenta al usuario final.

La comunicación se hace a través de redes, empleando un protocolo llamado HTTP.

Resumiendo, en la arquitectura cliente-servidor, el servidor almacena y provee información y el cliente tienen que presentar la información al usuario, empleando tecnologías de la web.

**Protocolos de comunicación:**

El protocolo de comunicación en Internet fue creado para la transferencia de archivos hipertextuales se llama HTTP, (HyerText Transfer Protocol)

Este protocolo tiene características a tener en cuenta:

1. Pedido-respuesta: se abre una conexión por cada pedido, que surge del cliente, y el servidor la cierra una vez enviada la respuesta.

2. Stateless: El protocolo de por si no maneja noción de memoria de pedidos anteriores

3. Textual: Se intercambian mensajes solo texto.

4. Basado en códigos de respuesta: Incluso para los flujos de error; no hay memoria compartida, continuaciones, excepciones ni eventos.

**Presentado:**

Los navegadores modernos son capaces de entender algunos lenguajes sin necesidad de ningún complemento, que son los que constituyen como el estándar de la web.

* HTML: Lenguaje basado en etiquetas, emparentado con XML, diseñado para estructurar información.
* CSS: Lenguaje para formatear información. (estructurada en HTML)
* JavaScript: Lenguaje de propósito genera, que en los navegadores es utilizado para desarrollar cualquier lógica de aplicación. Este ultimo nos permite entre otras cosas:
* Mutar, acceder a, y observar eventos del DOM (la representación del contenido de HTML)
* Implementar efectos visuales complejos; Realizar pedidos al servidor en segundo plano.
* Implementar navegabilidad del lado del cliente.

HTTP & REST:

**Primeros Pedidos:**

Cada recurso de la web es localizable gracias a un identificador único llamado URL. Las URL nos dan la ubicación de un recurso y como conseguirlo.

¿A que corresponde cada parte de la URL?

A picture containing text

Description automatically generated

Para hacer pedidos se usa la biblioteca requests. Se usa requests.get(“URL”) para traer lo que este en esa URL. Y luego una vez obtenido para mostrarlo se usa lo usado anteriormente con un “. json()”.

Text

Description automatically generated

El formato en que se devuelve estos valores es JSON. El formato tiene la característica de acoplar datos, se usa para el intercambio de datos, puede soportar distintos datos como: Strings, números, booleanos y null, aparte de objetos y arrays. Es de fácil lectura por su forma para humanos y se puede leer con python, JavaScript entre otros.

**Codigos de Respuesta:**

El atributo .headers sirve para traer todas las cabeceras de la respuesta. Es un diccionario especial que contiene cada una de las cabeceras devueltas como claves del diccionario. \

El atributo .status\_code devuelve el código de estado del request. Si la petición fue exitosa devuelve 200, sino encontró el URL solicitado devuelve 404. El código de respuesta **500 Error** Interno del Servidor del Protocolo de Transferencia de Hipertexto (HTTP) indica que el servidor encontró una condición inesperada que le impide completar la petición. Este código **es** una respuesta genérica. El código 201, quiere decir que fue exitosa la petición y que algo fue creado.

El atributo .content sirve para devolver el contenido cuando no es texto, como imagen o pdfs y los devuelve como una secuencia de bytes.

**Parámetros:**

Los parámetros sirven para “filtrar” el request desde la URL.

Se puede usar ?tipo=lo que quieras sacar, para filtrar por ese tipo de dato.

Text

Description automatically generated

Se puede agregar mas parámetros con &.

Text

Description automatically generated

**URLS Y URIS**

A <https://macowins-server.herokuapp.com/ventas/?_page=3> se lo suele llamar dirección pero el nombre técnico es URL. Una URL es cualquier string con un formato particular llamado URI, que nos permite localizar un recurso, en por ejemplo internet.

Las URI se conforman con:

* Un protocolo
* Un dominio
* Un puerto (opcional)
* Una ruta
* Parámetro(opcional)
* un fragmento, que indica en que sección queremos obtener del recurso que estamos consultando(opcional)

Este es un ejemplo “”protocolo://dominio:puerto/ruta#fragmento?parametro1=valor1&parametro2=valor2””

**Resolucion de dominios:**

Si se pide un dominio inexistente, es status code será 404, page not found.

El termino timeout, sirve para ejecutar esa acción durante n segundos, si la acción sigue tratando de realizarse post esos segundos tira error de timeout.

**Cabeceras:**

Las cabeceras son todas las cosas que tira el diccionario al hacer el atributo .headers. Estas son:

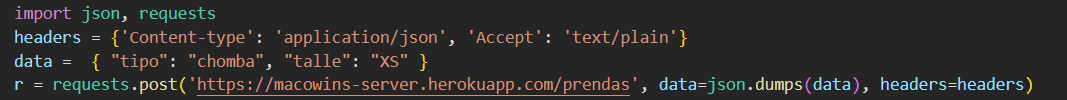
* **X-Powered-By**: QUIEN: nos dice que software es el que está corriendo en el servidor. No siempre es muy confiable
* **Content-Length**: TAMAÑO: nos dice el tamaño de la respuesta.
* **Date:** FECHA: nos dice la fecha en que se generó la respuesta.
* **Content-Type:** TIPO: nos dice el tipo de contenido que estamos recibiendo, el cual podría ser, por ejemplo:
  + sonido, como audio/wavo audio/ogg
  + video, como video/ogg
  + una imagen, como image/jpeg o image/gif
  + un XML application/xml
  + un archivo css text/css.
* **Content-encoding:** es usada para saber que codificación adicional ha sido aplicado al cuerpo de la entidad.
* Gzip
* Compress
* Deflate
* Identity
* Br
* **Transfer-encoding:** Especifica la forma de codificación utilizada para transferir datos de forma segura al cuerpo del playload.
* Chunked: Por fragmentos
* Compress
* Deflate
* Gzip
* Identity
* **Server:** Indica el servidor
* **Expire:** Indica la fecha y hora en la que se considera la respuesta caducada. Fechas invalidas, como “0” o “-1”, representan fechas pasadas.

**Creando y actualizando contenido.**

¿Qué es un método HTTP?

Es un conjunto de métodos de petición para indicar la acción que se desea realizar sobre un recurso.

Para agregar datos a una url, se puede hacer el método requests.post esto se hace con una data que se especifica en forma de diccionario y agregándola. Hay que hacerlo de la siguiente manera.



Para borrar se hace esto:

Text

Description automatically generated

**Recursos:**

Hay varios métodos para HTTP:

* OPTIONS: consultar meta-datos o configuraciones de HTTP
* GET: consultar un contenido sin efecto
* POST: crear un contenido
* PUT: actualizar de forma total un contenido
* PATCH: actualizar de forma parcial un contenido
* DELETE: eliminar un contenido

**Rest y sus URLs**

Rest es un tipo de arquitectura de desarrollo web que se basa en el correcto uso de URLs y HTTP.

* GET /ventas/: consultar todos (listar)
* DELETE /ventas/: borrar todos
* POST /ventas/: crear uno
* POST /ventas/1 crear uno con ID (QMP no lo soporta)
* GET /ventas/1: consultar uno
* PUT /ventas/1: modificar uno
* DELETE /ventas/1: eliminar uno

Existen reglas básicas para usar REST.

* Debe evitarse usar verbos
* No debemos tener más de una URI para identificar un mismo recurso
* Deben ser independiente de formato
* Deben mantener una jerarquía lógica
* Los filtrados de información de un recurso no se hacen en la URI