# 

# EJERCICIO 1

## Instalación del ambiente

El presente ejercicio busca realizar la instalación del ambiente para el desarrollo del trabajo práctico. A continuación se listará una serie de aplicaciones a instalar

1. **Instalar el IDE Visual Studio Code**: Un IDE (Entorno de desarrollo integrado), es una aplicación que nos brinda facilidades al momento de generar código. Dentro de VDFactory utilizamos el Visual Studio Code para los proyectos relacionados a aplicaciones web y mobile. El mismo permite trabajar con varios lenguajes tales como: HTML, CSS, C#, Javascript (Node, Angular, IONIC, React, Typescript, etc).
2. **Instalar GIT y GIT Bash**: Git es una aplicación utilizada para el control de versionado de código. En otras palabras, es una suerte de “disco” en donde se guardarán los distintos files que componen nuestra aplicación (ejemplo: index.html, estilos.css, etc). La gran ventaja de GIT es que al momento de subir una versión nueva de un archivo, genera un “backup” de la versión anterior para poder ser restaurado cuando se desee. Además, el código se encuentra en la “nube”, por lo que puede ser accedido y descargado desde cualquier lugar que tenga acceso a internet.
3. **Instalar Node.js**: Node.js es un lenguaje de programación del lado del servidor (backend).

Todas las herramientas son gratuitas, por lo que no habrá que lidiar con el problema del licenciamiento.

Recomendamos revisar los cursos de platzi de Node.js con las siguientes credenciales:

<https://platzi.com/>

mail: platzi@virtualdreams.io

user: virtualdreamsfactory

pass: Buffalo1Virtual

# EJERCICIO 2

Las siguientes preguntas están orientadas a la comprensión del protocolo HTTP. Son agnósticas al lenguaje de programación, la idea es comprender los conceptos del estándar:

1. ¿Qué es un servidor HTTP?
2. ¿Qué son los verbos HTTP? Mencionar los más conocidos
3. ¿Qué es un request y un response en una comunicación HTTP? ¿Qué son los headers?
4. ¿Qué es un queryString? (En el contexto de una url)
5. ¿Qué es el responseCode? ¿Qué significado tiene los posibles valores devueltos?
6. ¿Cómo se envía data en un Get y cómo en un POST?
7. ¿Qué verbo http utiliza el navegador cuando accedemos a una página?
8. Explicar brevemente qué son las estructuras de datos JSON y XML dando ejemplo de estructuras posibles.
9. Explicar brevemente el estándar SOAP
10. Explicar brevemente el estándar REST Full
11. ¿Qué son los headers en un request? ¿Para qué se utiliza el key Content-type en un header?}

1. Un servidor HTTP (Hypertext Transfer Protocol, o en español Protocolo de transferencia de hipertexto) es un programa informático que se comunica con un servidor web y se encarga del intercambio de éste con el navegador del usuario. El usuario realiza una petición, es decir realiza una acción o una búsqueda, y luego le devuelve una respuesta y cierra la conexión.

2. los http verbs son un conjunto de métodos de petición que indican la acción a realizar sobre un determinado recurso.

Existe una gran variedad de métodos, pero los más relevantes y utilizados son:

* GET: Se usa para consultar información.
* POST: Se utiliza para la creación de un nuevo registro.
* PUT: Se utiliza para actualiza todo un registro existente.
* PATCH: Es similar al put, solo que este actualiza solo un fragmento del registro.
* DELETE: Se utiliza para eliminar un registro existente.
* HEAD: Se utiliza para obtener información sobre un determinado recurso.

3. En http un request es la petición que se hace como cliente al servidor para obtener un recurso determinado. Por otro lado, un response es la respuesta que el servidor da a dicha petición.

Los headers llevan información necesaria para la comunicación Se utilizan principalmente para la coordinación entre el cliente y el servidor: se garantiza que el cliente pueda conocer la forma del archivo, saber si el archivo está actualizado y que el archivo cumpla con las expectativas del navegador.

4. QueryString es un término que se usa para hacer referencia a la interacción con una base de datos. Es la parte de la url que contiene los datos que deben pasar a las aplicaciones web.

5. Un responseCode es un código de estado de respuesta http que indica si la solicitud(request) se ha completado de forma exitosa o si tuvo algún error. Se agrupan en 5 clases: del 100 al 199 están las respuestas informativas, del 200 al 299 las satisfactorias, del 300 al 399 las redirecciones, del 400 al 499 los errores de los clientes, y del 500 al 599 los errores del servidor.

6. En un get se traen los datos que están en el servidor. Los datos que se envían al servidor son escritos en la misma dirección url. Estas llamadas pueden ser cacheadas (verse en el historial del navegador), indexadas por buscadores, o demás. Por lo cual podríamos decir que es una búsqueda de seguridad débil. Por otro lado, los post envían información desde el cliente para que sea procesada y actualice o  agregue información al servidor. Se deben introducir los parámetros que se desean enviar en la solicitud. Los datos envíados por este método no se quedan guardados en la caché ni en el historial de navegación.

7. Cuando ingresamos a una página nuestro navegador utiliza el verbo get para solicitar un recurso al servidor.

8. Las estructuras Json (JavaScript Object Notation) son un formato de texto sencillo para el intercambio de datos. Una de sus principales características es que es ligero y rápido, y puede ser utilizado prácticamente por todos los lenguajes de programación. Por otro lado, las estructuras XML (Extensible Markup Language) son un conjunto de códigos que se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas. El lenguaje XML proporciona una plataforma para definir elementos para crear un formato y generar un lenguaje personalizado. El diseño XML se centra en la simplicidad, la generalidad y la facilidad de uso y, por lo tanto, se utiliza para varios servicios web.

9. SOAP (Simple Object Access Protocol) es un protocolo ligero para el intercambio de información en entornos descentralizados y distribuidos. Son transmisiones de información de remitentes a destinatarios y se pueden combinar para crear patrones de petición/respuesta.

10. Un estándar rest (Representational State Transfer) es considerado una técnica de arquitectura de software, es decir, un conjunto de principios y patrones de comunicación que ayudan a crear y construir las APIs. Por otro lado, una API RESTful es un servicio que funciona como estándar para compartir información, en un sistema de doble vía: consulta y respuesta.

11. Los headers transmiten información que no se encuentra en el body de la request, datos como el navegador del cliente, el idioma, etc.

El content-type sirve para indicar al cliente qué tipo de contenido será retornado, así el navegador puede decidir cómo interpretar los contenidos.

# EJERCICIO 3:

Recomendamos previamente entender los conceptos de la sintaxis “json” antes de arrancar con los ejercicios.

Descargar el POSTMAN (aplicación para realizar request como cliente), adjuntando un screen de resolución para cada ítem:

1. Realizar un request GET a la URL: <https://reclutamiento-14cf7.firebaseio.com/personas.json>

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Realizar un request POST a la URL anterior, y con body:

{

"nombre":"Tu nombre",

"apellido":"Tu apellido",

"dni":11223322

}

**Tip:** (Marcar la opción “raw” como body)

Captura de pantalla de computadora

Descripción generada automáticamente

1. Realizar nuevamente un request GET a la URL: <https://reclutamiento-14cf7.firebaseio.com/personas.json> ¿Qué diferencias se observan?

Interfaz de usuario gráfica, Texto, Correo electrónico, Sitio web

Descripción generada automáticamente

La segunda vez que hice el get se cargaron mis datos en el archivo

# EJERCICIO 4:

Realizar un script en Node.js con nombre de página “mostrarPersonas.js” que haga un GET con la biblioteca **request-promise** a la url anterior: <https://reclutamiento-14cf7.firebaseio.com/personas.json> y muestre los resultados por consola.

# EJERCICIO 5:

Crear un servidor en Node.js con la biblioteca “express” que pueda recibir un request POST con el formato json previamente mencionado:  
Ej:

{

"nombre":"Tu nombre",

"apellido":"Tu apellido",

"dni":11223322

}

Se deberá validar que el json posea el formato correcto:

* El campo dni y apellido son obligatorios
* El campo nombre y apellido son cadenas de texto (String)
* El campo dni es numérico de máximo 10 caracteres
* Si el usuario manda un json con más atributos que los mencionados deberá notificarlo como error

Reciba la información deberá insertar en la BD realizando un post a la URL: <https://reclutamiento-14cf7.firebaseio.com/personas.json>

El servicio deberá responder:

status code: 201 y el json que devuelve el post en caso de que se haya ingresado exitosamente  
 status code 400 si el formato del json es inválido  
 status code 500 por cualquier otro error no previsto

# EJERCICIO 6:

Realizar un programa en HTML con nombre de página “crearPersonas.html” que tenga 3 inputs text en html (nombre, apellido y dni) y un botón de “enviar”. Al presionar dicho botón se deberá realizar un POST al servicio que fue levantando en el punto 5

# EJERCICIO 7:

Crear un repositorio de GIT y subir los resultados de los ejercicios. Añadir un README.md de como ejecutar el programa.