# 

# **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 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 estandar SOAP
10. Explicar brevemente el estandar REST Full
11. ¿Qué son los headers en un request? ¿Para qué se utilizar el key Content-type en un header?

# 

# **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 item:

1. Realizar un request GET a la URL: <https://reclutamiento-14cf7.firebaseio.com/personas.json>
2. 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)

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

**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 númerico 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

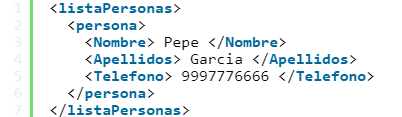
# **Ejercicio 6:**

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**

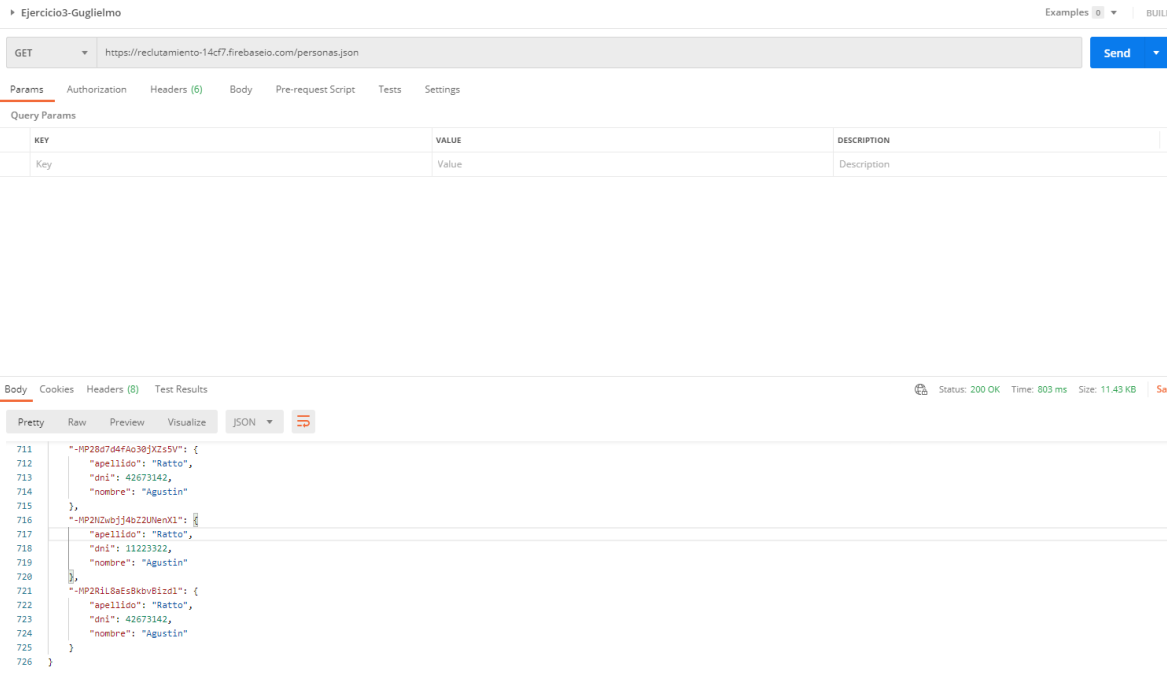
Crear un repositorio de GIT y subir los resultados de los ejercicios. Añadir un README.md de como ejecutar el programa.

**Respuestas Ejercicio 2:**

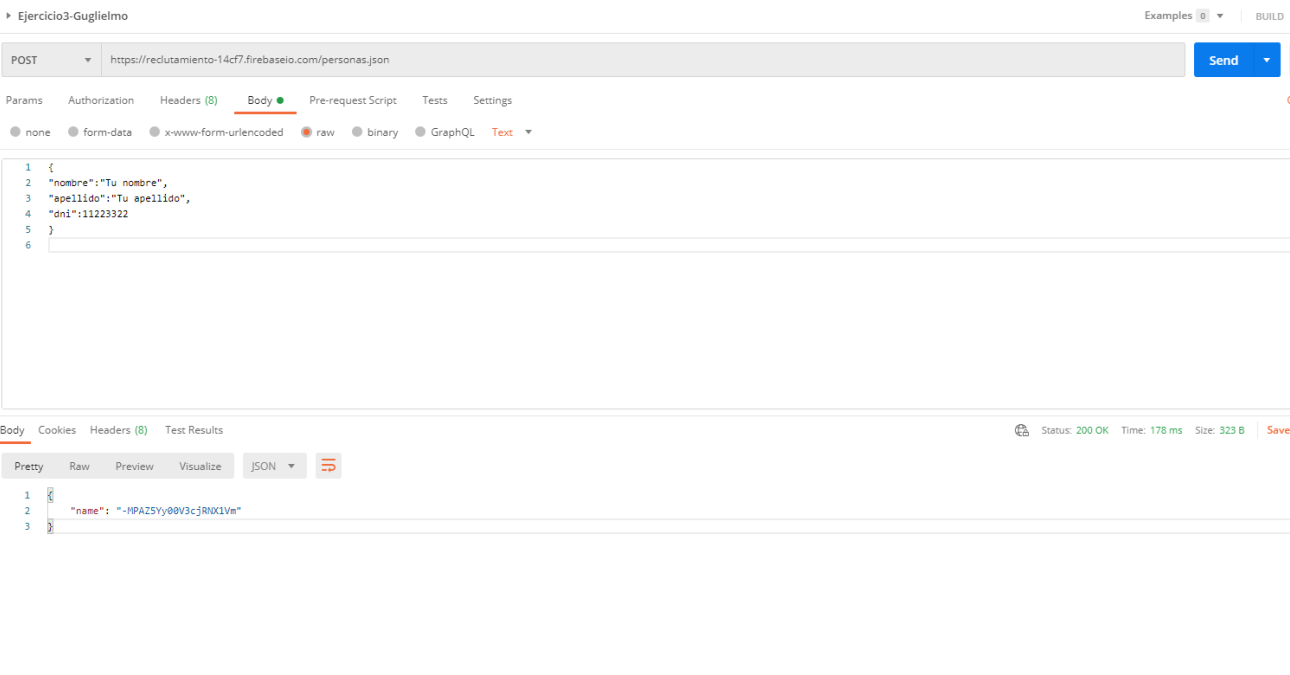
1. Un servidor HTTP (Hyper Text Transfer Protocool), o servidor web, es un programa informático que procesa una aplicación del lado del servidor. Se encarga de enviar peticiones desde la conexión del cliente hasta el sistema que se encarga de administrar cada tipo de petición, y, de este modo, se transmite la respuesta correspondiente de nuevo al cliente. En resumen, cuando introducimos una URL valida en nuestro navegador, la petición de conexión se envía al Servidor Web para administrar la petición.
2. Los verbos HTTP son un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Estos verbos son:  
   GET: solicita una representación de un recurso especifico (recuperan datos).  
   HEAD: pide una respuesta idéntica a la de una petición GET (sin el cuerpo de la respuesta).  
   POST: se utiliza para enviar una entidad a un recurso especifico (a menudo causa cambios).  
   PUT: reemplaza todas las representaciones actuales del recurso de destino con la petición.  
   DELETE: borra un recurso especifico.  
   CONNECT: establece un túnel hacia el servidor identificado por el recurso.  
   OPTIONS: utilizado para describir las opciones de comunicación.  
   TRACE: realiza una prueba de bucle de retorno de mensaje.  
   PATCH: utilizado para aplicar modificaciones parciales.
3. **Request** (solicitar) permite el acceso a toda la información que pasa desde el navegador del cliente al servidor. Recibe esta información, la reparte y almacena entre cinco colecciones. Una vez que se almacenaron, los datos pueden ser accedidos individualmente en cada colección a través de su clave.  
   Una vez que el navegador envía el HTTP request, el servidor responde con un **HTTP response**. Esta contiene el **protocolo**, **Status code**, que es el código de respuesta, por ej.: 200 OK, que significa que el GET request ha sido satisfactorio y el servidor devolverá los contenidos del documento solicitado. También contiene los **Headers**, que almacena la información sobre el software del servidor, cuando se modifico por ultima vez el resource solicitado, el mime type, etc.. Y por ultimo, el **body**, que devuelve información que no sean de headers.  
   Como bien se dijo, los **Headers HTTP** son los parámetros que se envían en una petición o respuesta HTTP al cliente o al servidor para proporcionar información esencial sobre la transacción en curso.
4. El Quiery String es la cadena de texto que se escribe en la ventana de URL del navegador y es un término informático que se utiliza para hacer referencia a una interacción con una base de datos.
5. El responseCode son los códigos de estado de una respuesta HTTP que indican si se ha completado satisfactoriamente una solicitud HTTP especifica. Hay 5 clases:  
   -Respuestas informativas (100-199)  
   -Respuestas satisfactorias (200-299)  
   -Redirecciones (300-399)  
   -Errores de los clientes (400-499)  
   -Errores de los servidores (500-599)
6. Los datos que se envían mediante **GET** van de forma “visible” al cliente. Por ej.: tenemos la URL  <http://www.aprenderaprogramar.com/newuser.php> **y, al utilizar el método GET, a modo de ejemplo, podría ser el siguiente:** [**http://www.aprenderaprogramar.com/newuser.php?nombre=Pepe&apellido=Flores&email=h52turam%40uco.es&sexo=Mujer**](http://www.aprenderaprogramar.com/newuser.php?nombre=Pepe&apellido=Flores&email=h52turam%40uco.es&sexo=Mujer). Donde después del ‘?’ aparecen parejas de datos con su nombre y valor separados por el símbolo &.  
   Mientras que POST consiste en datos “ocultos”, ya que el cliente no los ve. Por lo tanto, esto es mas seguro ya que el usuario no podría modificar la URL escribiendo diferentes parámetros a los reales en su navegador.
7. Cuando accedemos a una página usamos el método GET.
8. JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las maquinas es simple interpretarlo y generarlo. Es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores.  
   Esta constituido por dos estructuras:  
   -Una colección de pares de nombre/valor.  
   -Una lista ordenada de valores.  
   Comienza con ‘{‘ y termina con ‘}’, cada nombre es seguido por ‘:’ y los pares nombre/valor están separados por ‘,’.  
     
   XML (eXtensible Markup Languaje) es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos. Este conjunto de códigos, se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas. Se divide en 2 partes: *prolog* y *body. Prolog* consiste en metadatos administrativos, como declaraciones, instrucciones, declaración de tipo de documento y comentarios. *Body* se compone de dos partes: estructural y de contenido.  
   
9. SOAP es un protocolo estándar que se creo para permitir la comunicación entre las aplicaciones que se diseñaban con diferentes lenguajes y diferentes plataformas. Impone reglas integradas que aumentan la complejidad y sobrecarga, lo cual puede retrasar el tiempo que tardan las paginas en cargarse. Los estándares de cumplimiento integrados incluyen la seguridad, la atomicidad, la uniformidad, el aislamiento y la durabilidad.
10. REST es un conjunto de principios arquitectónicos que se ajusta a las necesidades de los servicios web y las aplicaciones móviles ligeros. Dado que se trata de un conjunto de pautas, la implementación de las recomendaciones depende de los desarrolladores. Cuando se envía una solicitud de datos a una API de REST se suele hace a través de un protocolo de transferencia de hipertexto (HTTP). Estas API pueden devolver mensajes en distintos formatos, el mas usado, es JSON. RESTful es más flexible y se puede configurar con mayor facilidad.  
      
    REST es un conjunto de pautas que ofrece implementación flexible, mientras que SOAP es un protocolo con requisitos específicos. Las API de REST son ligeras, asique son ideales para los contextos mas nuevos. Pero los servicios de web de SOAP ofrecen seguridad y cumplimiento de las operaciones integradas, pero que también los hacen más pesados.
11. Los HTTP headers son la parte central de los HTTP requests y responses, y transmiten información acerca del navegador del cliente, de la pagina solicitada, del servidor, etc.  
    El Content-Type es la propiedad del header usada para indicar el ‘media type’ del recurso. Le dice al cliente que tipo de contenido será retornado.

**Respuestas Ejercicio 3:**

1. GET



1. POST



1. Nuevamente, aplicamos un GET, que, a diferencia del anterior, ahora podemos ver que se ha ingresado nuestros datos al haber realizado un POST anteriormente.

