[**Ejercicio 1**](#_2v1gyw8tea5v) **2**

[Instalación del ambiente](#_duz18ghehv7o) 3

[**Ejercicio 2**](#_jjqjy3scwou) **3**

[**Ejercicio 3**](#_hhx3u2rgqypm) **4**

[**Ejercicio 4**](#_6gc3a8j82cb3) **5**

[**Ejercicio 6**](#_15yg8oman8qz) **5**

# **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:**

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:

2.1. un servidor HTTP es un programa informático que como su nombre lo indica ejecuta la una aplicación del lado servidor, encargándose e las conexiones con el cliente y generando una respuesta para la aplicación del lado del cliente.

2.2. los HTTP VERBS son un conjunto de métodos que indican la acción a realizar con un recurso determinado.

Algunos de los más utilizados son:

* GET: solicita la representación de un recurso especifico.
* POST: se utiliza para la “creación” de nuevos recursos.
* PUT: actualiza un recurso determinado.
* DELETE: como su nombre lo indica, elimina un recurso.

2.3. En HTTP un REQUEST es la petición que se hace como cliente al servidor para obtener un recurso determinado. En cambio, un RESPONSE es la respuesta que el servidor da a dicha petición.

La estructura de un REQUEST es la siguiente:

* Método: GET, POST, PUT, etc. que indica que tipo de RECUEST es.
* PATH: indica donde se encuentra el recurso.
* Protocolo: indica que protocolo se va a usar y su versión, por ejemplo, HTTP/1.1
* HEADERS: son esquemas de KEY VALUE que contienen información sobre el navegador del cliente e información sobre el REQUEST en general.
* BODY: si se envía información al servidor a través de POST y PUT, esta va en el BODY.

La estructura de un RESPOSE es la siguiente:

* Protocolo: al igual que en una REQUEST contiene el protocolo y la versión de este.
* STATUS CODE o RESPONSECODE: es un código que se da como repuesta a la REQUEST, por ejemplo 404, que indica que el recurso no fue encontrado.
* HEADERS: Contiene información sobre el software del servidor, la fecha de modificación del recurso, etc.
* BODY: si se devuelve información que no esté incluida HEADER, esta se ubica en el BODY.

2.4. QUERYSTRING es un término que se usa para hacer referencia a la interacción con una base de datos. Son parte de una URL y a través de estas se pueden pasar parámetros las aplicaciones WEB

2.5. Un RESPOSECODE es un código que se da como respuesta a una REQUEST, los códigos devueltos dan información sobre el resultado de la REQUEST, pueden indicar si fue exitosa, si hubo algún error, y si lo hubo con el código podemos identificarlo fácilmente.

2.6 En los GET y en los POST la información se envía en HEADER y el BODY.

El HEADER contiene información general acerca del cliente o servidor (dependiendo si es un REQUEST o un RESPONCE) y el BODY contiene del mensaje en sí.

2.7 cuando ingresamos a una página nuestro navegador utiliza el verbo GET para solicitar un recurso al servidor.

2.8 JSON (JavaScript Object Notation) es un formato de texto sencillo para el intercambio de datos. Debido a su amplia aceptación como alternativa a XML se lo considera un formato independiente a su lenguaje.

{ “Info”:

{

“Empleados”:[

{

“name”: “Maximiliano Caseres”,

“age”:21,

}

{

“name”:”Juan Perez”,

“age”:25,

}

]

}

}

XML (eXtensible Markup Language), proviene del lenguaje SGML y a diferencia de otros lenguajes, XML da soporte a bases de datos, siendo muy útil cuando varias aplicaciones deben comunicarse entre sí o intercambiar información.

<Info>

<Empleados>

<Empleado>

<name>Maximiliano Caseres</name>

<age>21<age>

</Empleado>

<Empleado>

<name></name>

<age></age>

</Empleado>

</Empleados>

</Info>

2.9. SOAP (Simple Object Access Protocol) es un protocolo estándar que define como dos objetos de diferentes procesos pueden comunicarse a través del intercambio de datos XML.

2.10. REST (REpresentational State Transfer) es considerada una técnica de arquitectura de software, es decir, un conjunto de principios y patrones de comunicación que ayudan a crear y construir las APIs.

Una API RESTful es un servicio que funciona como estándar para compartir información, en un sistema de doble vía: Consulta y Respuesta (REQUEST => RESPONSE)

2.11. Los HTTP HEADERS transmiten información que no se encuentra en el BODY de la REQUEST, datos como el navegador del cliente, idioma, etc.

El CONTEN-TYPE sirve para indicar al cliente que tipo de contenido será retornado, así el navegador puede decidir cómo interpretar los contenidos.

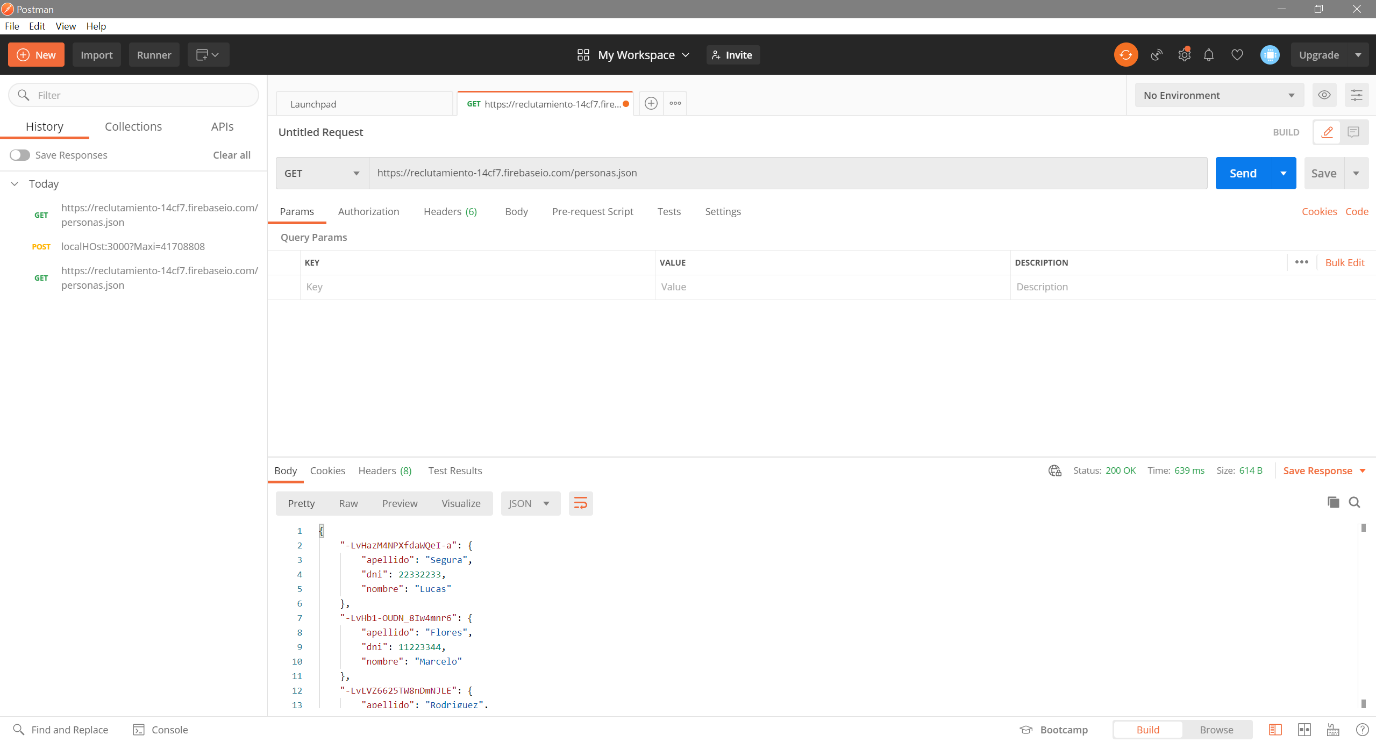
Ejemplos:

Content-Type: text/html; charset=UTF-8

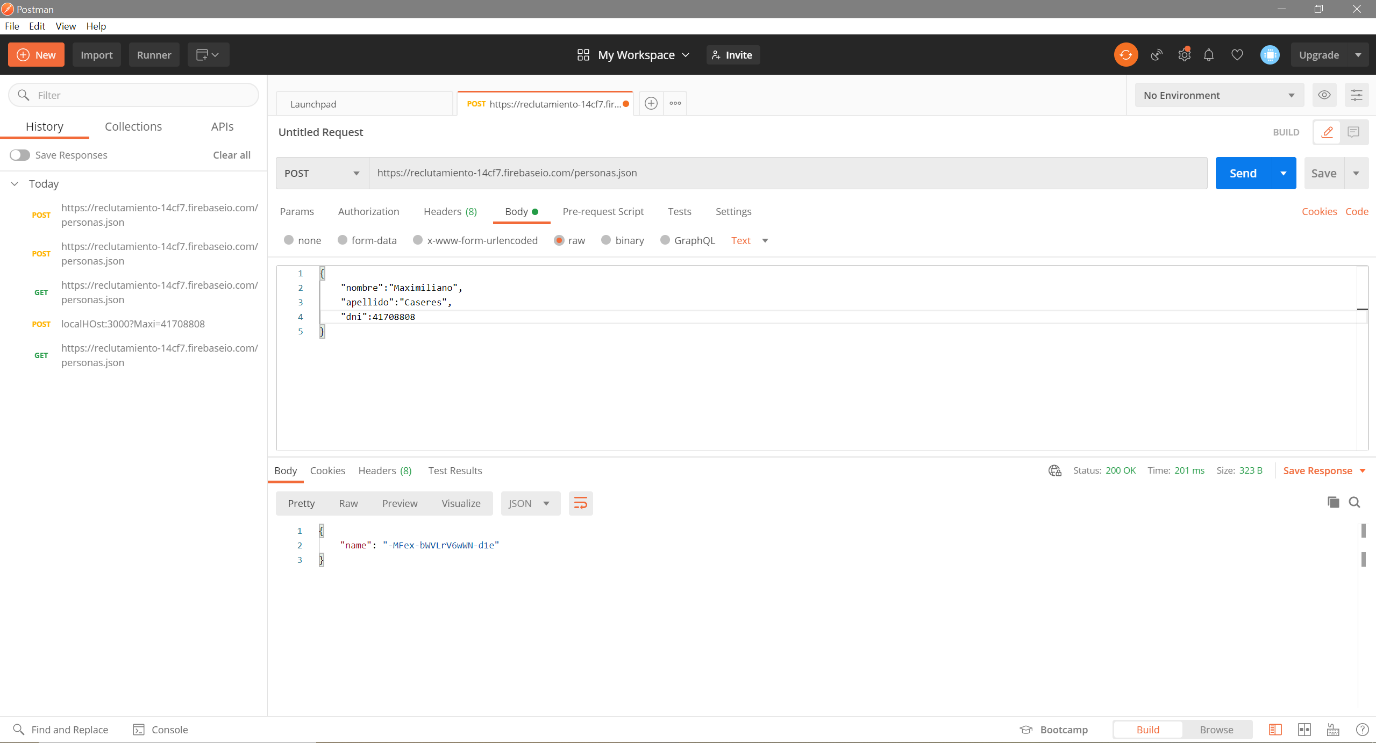
Content-Type: image/gif

Content-Type: application/pdf

3. Primer GET:



POST:



Segundo GET: puedo ver que se agregaron los datos que envíe anteriormente

