# 

# 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: VDFactory

# 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?

# 

# 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>
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 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

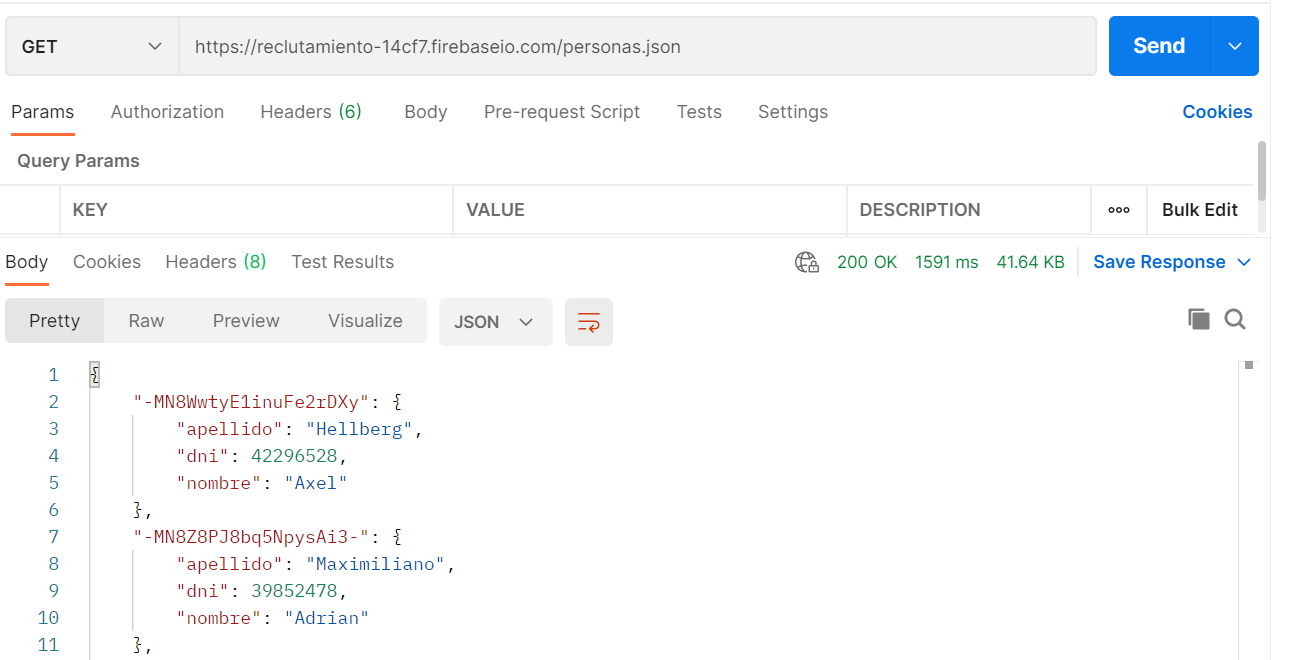
# EJERCICIO 6:

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

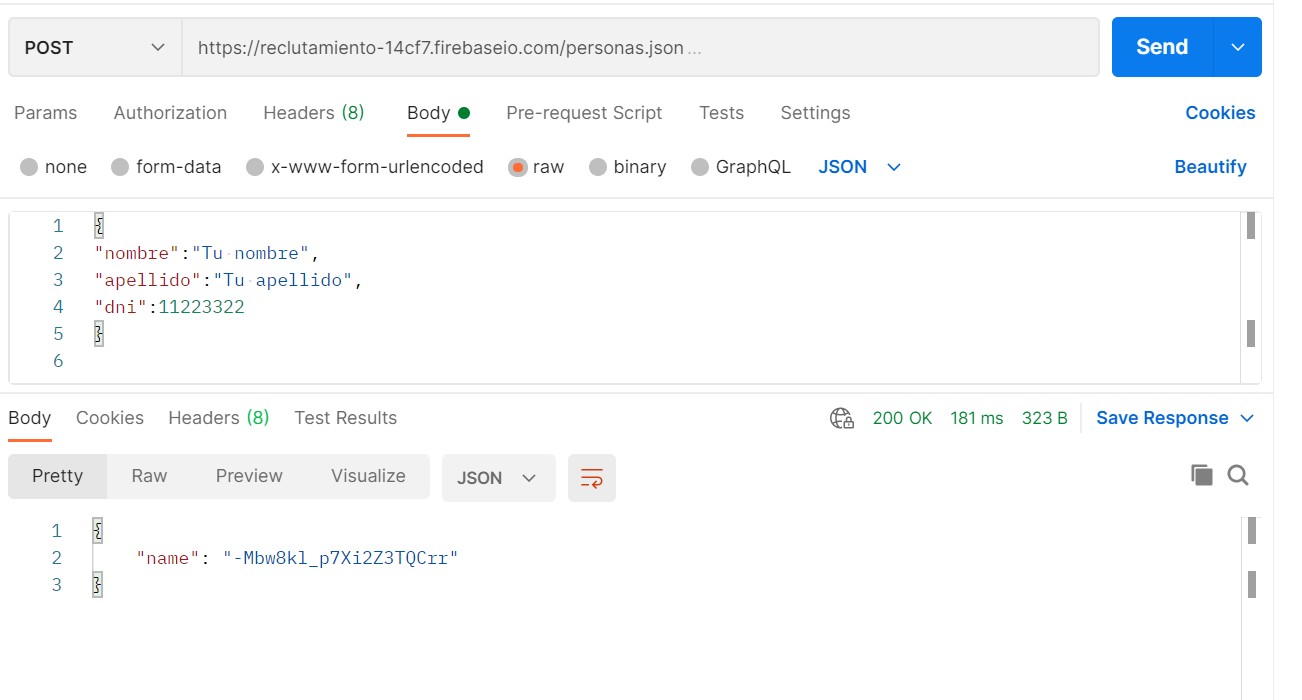
# EJERCICIO 7:

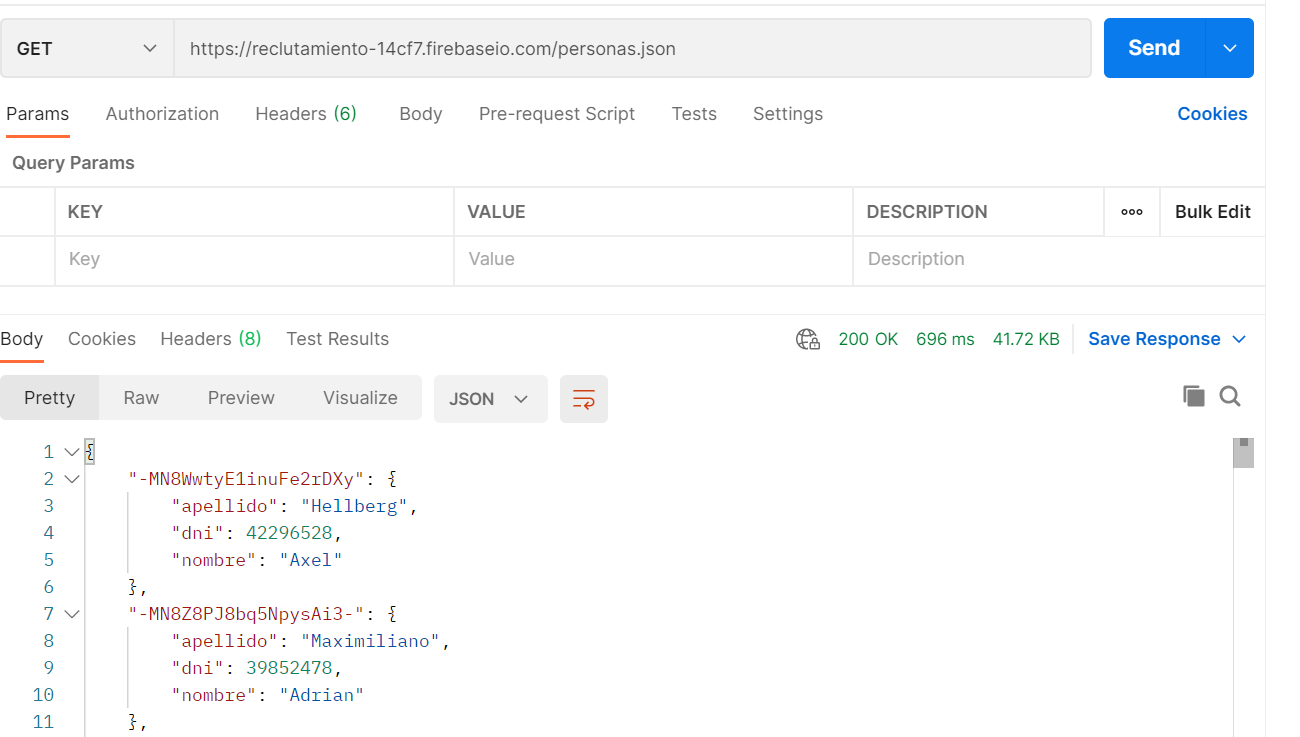
Crear un repositorio de GIT y subir los resultados de los ejercicios. Añadir un README.md de como ejecutar el programa y las respuestas del ej2.

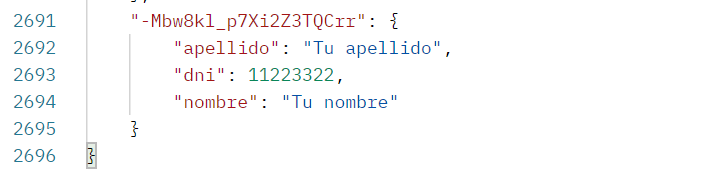
Ejercicio 3:

1.

2.



3. 

La diferencia es que en el segundo GET se puede observar que al final de la lista se agregó un nuevo objeto cuya clave corresponde con la respuesta del método POST anteriormente utilizado.