DESARROLLO DE APLICACIONES WEB AVANZADO

LABORATORIO N° 03

Sitio Web con Express

Alumno(s):		***Nota***	
Grupo:		***Ciclo:V***	

Criterio de Evaluación	*Excelente (4pts)*	*Bueno (3pts)*	*Requiere mejora (2pts)*	*No acept. (0pts)*	*Puntaje Logrado*
Entiende cómo se declaran los middlewares					
Hace testing de métodos HTTP					
Desarrolla aplicaciones web					
Realiza con éxito lo propuesto en la tarea.					
Es puntual y redacta el informe adecuadamente					

Laboratorio 3:
Sitio Web con Express

Objetivos:

Al finalizar el laboratorio el estudiante será capaz de:
- Entender el funcionamiento de los middlewares
- Desarrollar aplicaciones web con el framework Express
- Resolución de problemas en Javascript

Seguridad:

- Ubicar maletines y/o mochilas en el gabinete del aula de Laboratorio.
- No ingresar con líquidos, ni comida al aula de Laboratorio.
- Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

Equipos y Materiales:

- Una computadora con:
  - Windows 7 o superior
  - VMware Workstation 10+ o VMware Player 7+
  - Conexión a la red del laboratorio
- Máquinas virtuales:
  - Windows 7 Pro 64bits Español - Plantilla
- Instalador de node.js

Procedimiento:

Lab Setup

1. **Creación del equipo virtual**
   1.1. Encender el equipo
   1.2. Acceder empleando la cuenta de **usuario**: Tecsup, **contraseña**: _____
   1.3. Iniciar el Software VMWare.
   1.4. Abrir la plantilla ubicada en:
   E:\Equipos virtuales\Windows 7 Pro 64bits Español - Plantilla
   1.5. Crear un clon de la maquina anterior con los siguientes datos:
   1.6. Nombre del clon: **C15-DAWA**
   1.7. Ubicación del clon: **D:\C15-DAWA**
   1.8. Cerrar la plantilla
   1.9. Iniciar el equipo virtual **C15-DAWA**
   1.10. Identifíquese con la cuenta de usuario: **Redes**. Contraseña: **RCDTecsup2**
   1.11. Instale **node.js**

Entendimiento de ejecución de funciones en JavaScript

2. Iniciaremos la creación de nuestro laboratorio 3 creando la carpeta de nuestro proyecto en la raíz de la unidad C:
   2.1. Ingresamos los siguientes comandos y seguimos el asistente de NPM

```
a>mkdir lab03

a>cd lab03

a\lab03>npm init
```

2.2. Instalamos la librería Express de modo local.

```
>npm install express --save
```

2.3. Creamos el archivo index.js con el siguiente contenido.

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hola mundo! en Express :)')
})

app.listen(3000, function () {
  console.log('Aplicacion de ejemplo escuchando en el puerto 3000!')
});
```

2.4. Como vemos, el código es distinto pero un poco más intuitivo. Esto es lo básico para nuestra aplicación en Express. Sin embargo, generaremos un esqueleto para conocer cuál es el estándar recomendado para Express.

2.5. Instalaremos la herramienta express-generator, de manera global.

```
>npm install express-generator -g
```

2.6. Mostramos todas las opciones disponibles de express con el argumento –-help o -h

```
>express -h
```

2.7. Por ejemplo, crearemos la aplicación llamada **miapp**. Esta aplicación será creada en un folder llamado **miapp** en el directorio actual de trabajo.

```
>express miapp
```

2.8. Luego instalamos las dependencias (regeneramos el archivo npm)

```
>cd miapp

\miapp>npm install
```

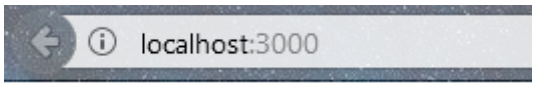
2.9. Si estas usando MacOS o Linux, ejecuta el siguiente comando:

```
>>DEBUG=miapp:* npm start
```

2.10. Si estas usando Windows, utiliza este:

```
>set DEBUG=miapp:* & npm start
```

2.11. Luego abre en el navegador la siguiente dirección:



2.12. La estructura de la aplicación creada por el generador es una de las muchas formas que se puede estructurar una aplicación Express. Puedes usar este modelo o el que desees.

Ahora reemplaza el archive index.js que se creó en el inicio del laboratorio por el siguiente.

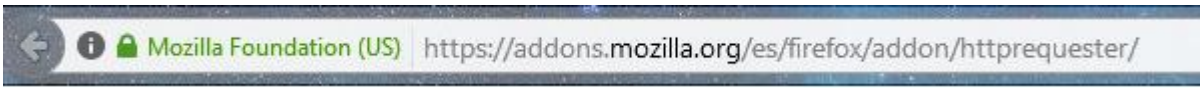
```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hola mundo! en Express :)')
})

app.post('/', function (req, res) {
  res.send('Llamada POST misma url')
})

app.put('/user', function (req, res) {
  res.send('Recibimos un PUT en /user')
})

app.delete('/user', function (req, res) {
  res.send('Recibimos un DELETE en /user')
})

app.listen(3000, function () {
  console.log('Aplicacion de ejemplo escuchando en el puerto 3000!')
});
```

2.13. Explica los cambios que se buscan al utilizar middlewares para capturar la misma URL pero con diferente método HTTP.

_____
_____
_____

2.14. Instala el navegador Mozilla Firefox. Este siempre ha sido un navegador con muchos complementos para los desarrolladores web.

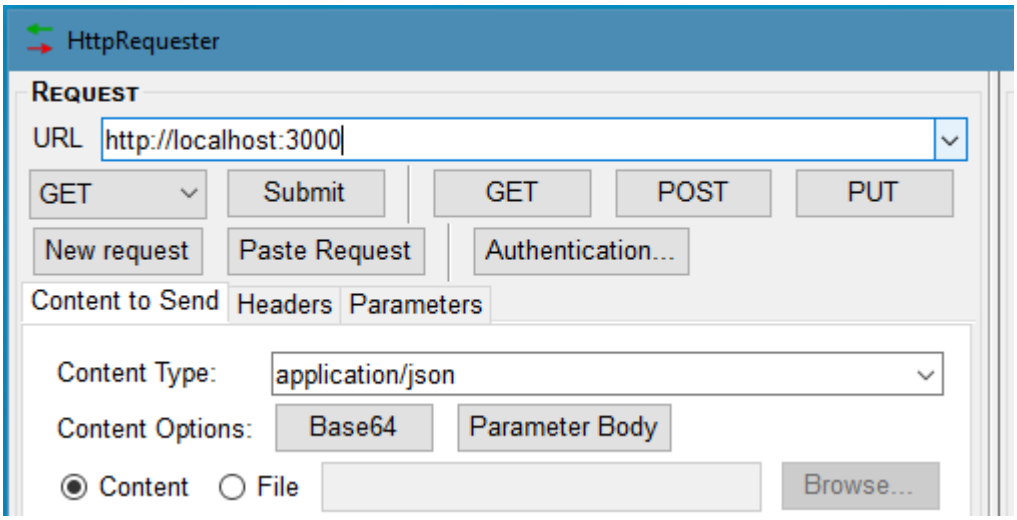
2.15. Ingresa a la siguiente dirección:



2.16. Estamos accediendo al complemento HttpRequester. Instalelo y reinicie el navegador cuando le pida la aplicación.

2.17. Se habrá añadido un ícono con el complemento en la parte superior derecha. Dele click para abrir la herramienta.

2.18. Para utilizar la herramienta, no basta más que con ingresar la URL deseada y hacer click en el método HTTP que deseemos probar. Adjunte capturas del uso de los middlewares declarados en el **index.js**



2.19. Para agregar archivo estáticos como imágenes, archivos de estilo CSS, y archivos JavaScript, use la función middleware **express.static**.
Para hacer esto, basta con pasar el nombre del directorio que contiene los archivos que deseamos publicar. Genere la siguiente estructura con archivos de ejemplo (pueden estar vacios o contener algo para demostrar el funcionamiento de la aplicación)

2.20. Declárelos con la función antes mencionada.

```
app.use(express.static('public'))
```

2.21. Pruebe la carga de estos archivos accediendo a sus URLs respectivas. Adjunte capturas de su verificación.

2.22. Express permite el uso de varios directorios. Basta con usar otro middleware parecido, no hay límite de uso.

```
app.use(express.static('public'))
app.use(express.static('files'))
```

2.23. También se pueden crear prefijos de rutas, para el enmascaramiento de archivos. Basta con pasarle un argumento de prefijo para lograrlo.

```
app.use('/static', express.static('public'))
```

2.24. Adjunte capturas de la verificación de la existencia de los archivos.

2.25. Para manejar errores necesitamos hacer uso de un middleware de errores. Primero lidiaremos con uno que devuelva algo cuando se accede a un recurso que no hemos declarado, es decir, “no existe”. Ya que los middlewares se ejecutan de manera progresiva, es importante decir que para el uso de un middleware de código 404 debe hallarse por lo general casi al final de su código.

```
app.use(function (req, res, next) {
  res.status(404).send("Eso no existe!")
})
```

2.26. La única diferencia será el middleware que capturará excepciones de código. Este recibirá un parámetro adicional, que es el error sucedido.

```
app.use(function (err, req, res, next) {
  console.error(err.stack)
  res.status(500).send('Algo salio ma!')
})
```

3. Inicio de proyecto

3.1. Elabore la landing-page de su proyecto. Cada integrante presentará su propuesta de trabajo y se elegirá la más adecuada para su emprendimiento. Deberán usar lo desarrollado en el laboratorio, todo gestionado por Express. Solicite a su instructor sus credenciales para publicar su proyecto.

4. **Finalizar la sesión**
   4.1. Apagar el equipo virtual
   4.2. Apagar el equipo

Conclusiones:

Indicar las conclusiones que llegó después de los temas tratados de manera práctica en este laboratorio.