UNIVERSIDAD POLITÉCNICA SALESIANA

CARRERA: COMPUTACIÓN
ASIGNATURA: PLATAFORMAS WEB
PRACTICA: 03
TÍTULO: BASES DE NODE.JS
Nombre: Eduardo Santiago Quisupangui Lema

OBJETIVO:

- Entender las bases de Node.js

PRE REQUISITOS:

a) Computador con Microsoft Windows o GNU/Linux
b) Interprete de Node.js versión 12 o superior
c) Editor de código fuente Visual Studio Code
d) Repositorio de software Git

INSTRUCCIONES:

1. Lea detenidamente cada uno de los enunciados propuestos
2. Plantee una solución a cada uno de los ejercicios
3. Programe una solución utilizando el lenguaje de programación Node.js
4. Elabore un informe con la solución de los ejercicios

ACTIVIDADES A DESARROLLAR:

1. **Documentar scripts**

Estudiar, ejecutar y documentar (colocar comentarios) los scripts revisados en clase. Por cada uno de los ejercicios, colocar una descripción en sus propias palabras de los conceptos más importantes. El informe debe contener los siguientes ejercicios:

- Requerir paquetes

Colocamos lo siguiente comando (npm init) para poder tener nuestros paquetes

```
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\03baseNode> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (03basenode)
```

Resultado:

package.json

```
{
    "name": "03basenode",
    "version": "1.0.0",
    "description": "Aplicacion para multiplicar",
    "main": "app.js",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "author": "Eduardo",
    "license": "ISC",
    "dependencies": {
        "colors": "^1.4.0",
        "yargs": "^15.3.1"
    },
    "devDependencies": {}
}
```

Para obtener los paquetes de colores colocamos el
siguiente comando e instalamos (npm install colors yargs –dave)

```
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\03baseNode> npm install colors yargs --save
[...................] / rollbackFailedOptional: verb npm-session d1e87475afa94b6f
```

Resultado:

```
    },
    "author": "Eduardo",
    "license": "ISC",
    "dependencies": {
        "colors": "^1.4.0",
        "yargs": "^15.3.1"
    },
    "devDependencies": {}
}
```

- Importar archivos al proyecto

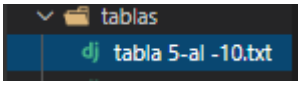
  Creamos una promesa para que así poder realizar un corrido y hacer la multiplicación en eso generar un archivo y se guarde como un archivo .txt

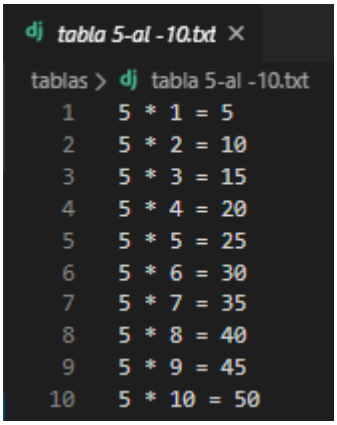
```
//CREAR UNA PROMESA
let crearArchivo = (base, limite = 10) => {
        return new Promise((resolve, reject) => {
            //PROMESA
            //validar que la base dea un numero
            if (!Number(base)) {
                reject(`el valor de la base ${base} no es valido`);
                return;
            }
            let data = ''; //contiene la informacion

            for (let i = 1; i <= limite; i++) {
                data += `${base} * ${i} = ${base * i}\n`; //para imprimir en un archivo
            }
            fs.writeFile(`tablas/tabla ${base}-al -
${limite}.txt`, data, (err) => { //aumento fs es un objeto
                //el path es relativo
                //se crear un archivo .txt y se dirige a la carpeta tablas
                if (err)
                    reject(err);
                else
                    resolve(`el archivo se guardo de la tabla ${base} al ${limite}`);
            });
        });
    }
    //PARA EXPORTAR UNA FUNCION
```

Resultado:

```
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\03baseNode> node .\app.js crear -b 5
el archivo se guardo de la tabla 5 al 10
```





- Recibir información de la línea de comandos

Para recibir información en las líneas de comando se realiza un menú con los alias y la ayuda.

```
//hacemos una variable
// opc es un objeto que tiene dos atributos base y limite
let opc = {
    base: {
        demand: true,
        alias: 'b'
    }, //creamos otro argumento o alias para si poder imprimir
    limite: {
        alias: 'l',
        default: 10
    }
}

const argv = require('yargs') //toca configurar los comandos
    .command('listar', 'imprime en consola la tabla de multiplicar', opc)
    .command('crear', 'crear un archivo de la tabla de multipplicar', opc)
    .help()
    .argv;
//yargs.js se hace un modulo ai¡si que tengoque exportar
module.exports = {
    argv
}
```

Resultados:

Al no saber que comando podemos colocar nos aparece la ayuda en la consola

```
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\03baseNode> node .\app.js crear -l
app.js crear

crear un archivo de la tabla de multipplicar

Options:
  --version     Muestra número de versión                          [booleano]
  --help        Muestra ayuda                                      [booleano]
  --base, -b                                                      [requerido]
  --limite, -l                                                  [defecto: 10]

Falta argumento requerido: base
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\03baseNode>
```

Al ingresar correctamente los comandos no sindica el mensaje de correcto.

```
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\03baseNode> node .\app.js crear -b 5 -l 15
el archivo se guardo de la tabla 5 al 15
```

Se guarda un archivo .txt

tabla 5-al -15.txt

tablas > tabla 5-al -15.txt

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
5 * 11 = 55
5 * 12 = 60
5 * 13 = 65
5 * 14 = 70
5 * 15 = 75
```

- Manejo de paquetes

Al manejar los paquetes podemos realizar diferentes acciones como usar los alias para usar por medio un menú de opciones y en eso podemos colocar colores para una presentación adecuado.

```
{
    "name": "03basenode",
    "version": "1.0.0",
    "description": "Aplicacion para multiplicar",
    "main": "app.js",
    "scripts": {
```

```
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "author": "Eduardo",
    "license": "ISC",
    "dependencies": {
        "colors": "^1.4.0",
        "yargs": "^15.3.1"
    },
    "devDependencies": {}
}
```

- Yargs

Yargs ayuda a crear herramientas interactivas de línea de comandos, analizando argumentos y generando una elegante interfaz de usuario.

```
//hacemos una variable
// opc es un objeto que tiene dos atributos base y limite
let opc = {
    base: {
        demand: true,
        alias: 'b'
    }, //creamos otro argumento o alias para si poder imprimir
    limite: {
        alias: 'l',
        default: 10
    }
}

const argv = require('yargs') //toca configurar los comandos
    .command('listar', 'imprime en consola la tabla de multiplicar', opc)
    .command('crear', 'crear un archivo de la tabla de multipplicar', opc)
    .help()
    .argv;
//yargs.js se hace un modulo ai¡si que tengoque exportar
module.exports = {
    argv
}
```

- Configuración YARGS

Se configura los comandos como en el código tenemos (listar y crear) en listar imprime la tabla de multiplicar y en crear genera un archivo con la tabla de multiplicar

```
let opc = {
    base: {
        demand: true,
        alias: 'b'
    }, //creamos otro argumento o alias para si poder imprimir
    limite: {
        alias: 'l',
        default: 10
    }
}

const argv = require('yargs') //toca configurar los comandos
    .command('listar', 'imprime en consola la tabla de multiplicar', opc)
    .command('crear', 'crear un archivo de la tabla de multipplicar', opc)
    .help()
    .argv;
```

- Optimización de Yargs

Se crea un menú para poder elegir las opciones de crear y listar utilizando los comandos de yargs

```
//tema de yargs
var colors = require('colors/safe');
//llamo a listartabla, creararchivo y argv
const argv = require("./config/yargs").argv;

const { crearArchivo, listarTabla } = require('./multiplicar/multiplicar');
//console.log(argv.base, argv.limite);
let comando = argv._[0];
let base = argv.base;
let limite = argv.limite;

switch (comando) {
    case 'listar':
        //llamamos la funcion listar tabla nos necesario poner
        //el then y el catch ya que es una funcion simple
        listarTabla(base, limite);
        break;
    case 'crear':
```

```
        crearArchivo(base, limite)
            .then(mensaje => console.log(colors.blue(mensaje)))
            .catch(err => console.log(colors.red(err)));
        break;
    default:
        console.log('comando no valido!');

}
```

- Colores en la Consola

Permite visualizar mejor los mensajes en consola como de respuesta y error

```
case 'crear':
       crearArchivo(base, limite)
           .then(mensaje => console.log(colors.blue(mensaje)))
           .catch(err => console.log(colors.red(err)));
       break;
```

Resultado:
Mensaje de respuesta de crear:

```
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\03baseNode> node .\app.js crear -l 3 -b 3
el archivo se guardo de la tabla 3 al 3
```

Mensaje de respuesta de listar:

```
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\03baseNode> node .\app.js listar -l 3 -b 3
========================
tabla de multiplicar 3
========================
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
```

- Publicar proyecto en GitHub

https://github.com/Eduardo-Quisupangui/03-node-js

2. GitHub

RESULTADOS OBTENIDOS:
1. El estudiante está familiarizado con la sintaxis y los fundamentos del lenguaje Node.js

CONCLUSIONES:

- Al manejar yargs nos ayudó mucho para poder llamar a las diferentes funciones del programa ya que nos permitió crear comandos, así poder ejecutar en consola la aplicación de la multiplicación.

REFERENCIAS:
[1] F. OpenJS, “Node.js,” nodejs.org. [Online]. Available: https://nodejs.org/en/.