PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE

IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE

- **Denominación del Programa de Formación:** TGO Análisis y Desarrollo de Sistemas de Información
- **Código del Programa de Formación:** 228106 V102
- **Nombre del Proyecto:** Construcción de un sistema de información que cumpla con los requerimientos del cliente en procesos que se lleven a cabo en el sector productivo del departamento de Caldas
- **Fase del Proyecto:** IMPLEMENTACIÓN
- **Actividad de Proyecto:** Seleccionar la alternativa de solución que cumpla con los requerimientos establecidos por el cliente
- **Competencia:** Construir el sistema que cumpla con los requisitos de la solución informática.
- **Resultados de Aprendizaje Alcanzar:** Realizar la codificación de los módulos del sistema y el programa principal, a partir de la utilización del lenguaje de programación seleccionado, de acuerdo con las especificaciones del diseño
- **Duración de la Guía:** 40 horas

2. PRESENTACIÓN

En esta guía los aprendices con el acompañamiento del instructor, desarrollarán ejemplos y proyectos utilizando herramientas tecnológicas de vanguardia con la arquitectura **API Rest**. Se desarrollará una aplicación que permita desarrollar habilidades específicas en el área Web integrando diversos componentes inicialmente desde el área del Backend.

Rest: Representational State Transfer
API: Application Programming Interface

.

3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

- **Descripción de la(s) Actividad(es)**
  - **Actividades de aprendizaje**
    - Creación de Servicios para comunicación con el Backend
    - Utilización del ruteo (Routing) para movernos entre los diferentes componentes
    - Creación de Formularios
    - Creación de API Rest utilizando Express y Node
    - Utilización de base de datos para la persistencia de datos

- **Actividad de Reflexión inicial**

Actividad de reflexión inicial

Antes de comenzar los diferentes proyectos, los aprendices deberán conocer la terminología básica de API Rest, complementando los conceptos fundamentales que se han visto en las guías pasadas. Para ello se deberán analizar los siguientes videos:

¿Que son las APIs y para qué sirven?
https://www.youtube.com/watch?v=u2Ms34GE14U

REST y RESTful APIs | Te lo explico en 5 minutos!
https://www.youtube.com/watch?v=JD6VNRdGl98

- **Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje**

Preparación inicial

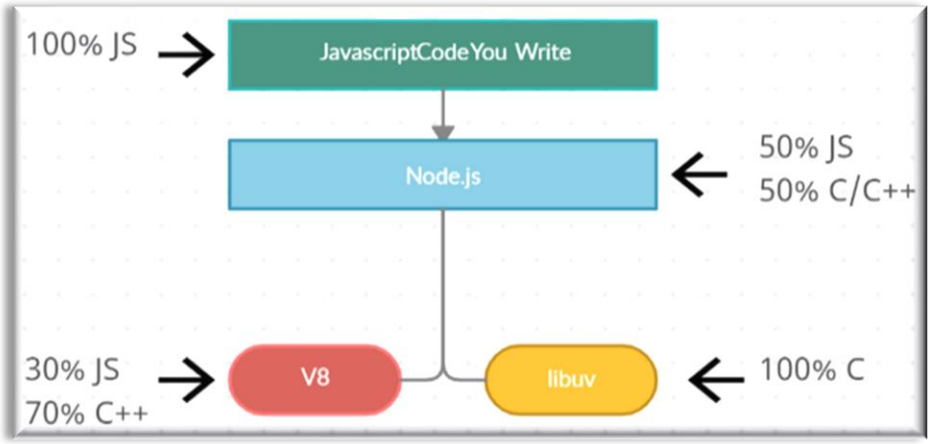
Para el desarrollo de los ejemplos y aplicación, primero veremos algunos detalles y conceptos fundamentales de las plataformas que se usarán, y que debemos instalar en nuestros equipos.

Node.js	
https://nodejs.org/	

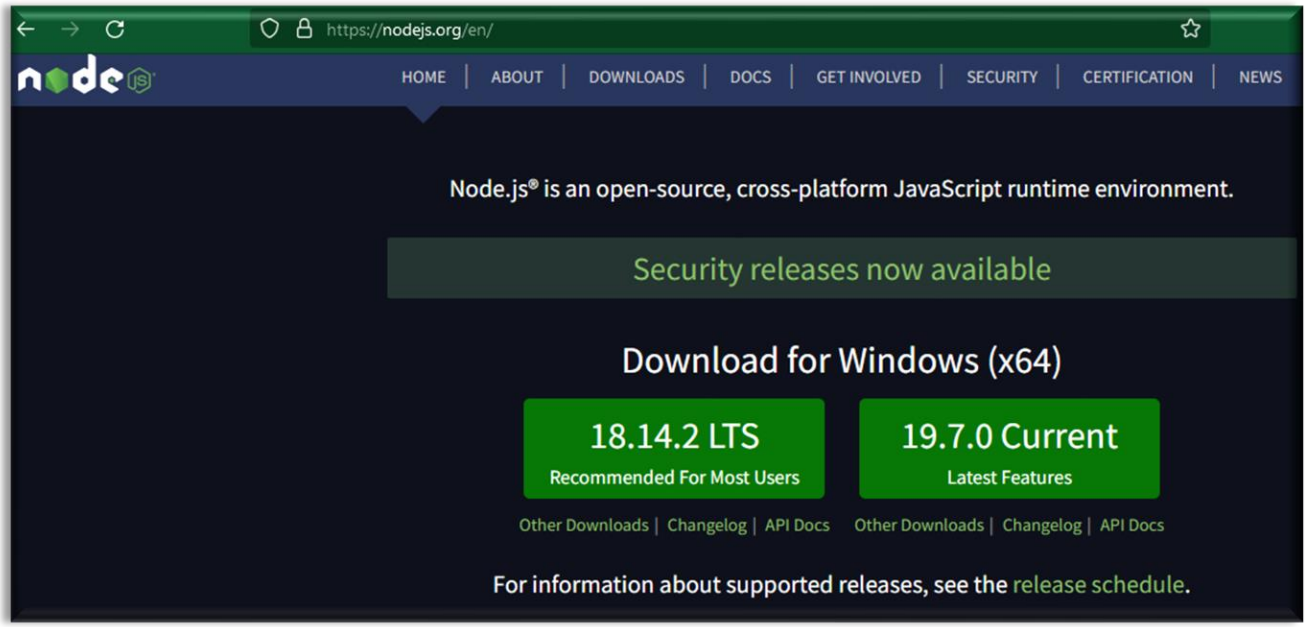
Node.js es un entorno de ejecución de Javascript creado en 2009 y está orientado a servidores, de código abierto, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor **V8 de Google**. Es una de las formas más rápidas y escalables para correr código desde el servidor

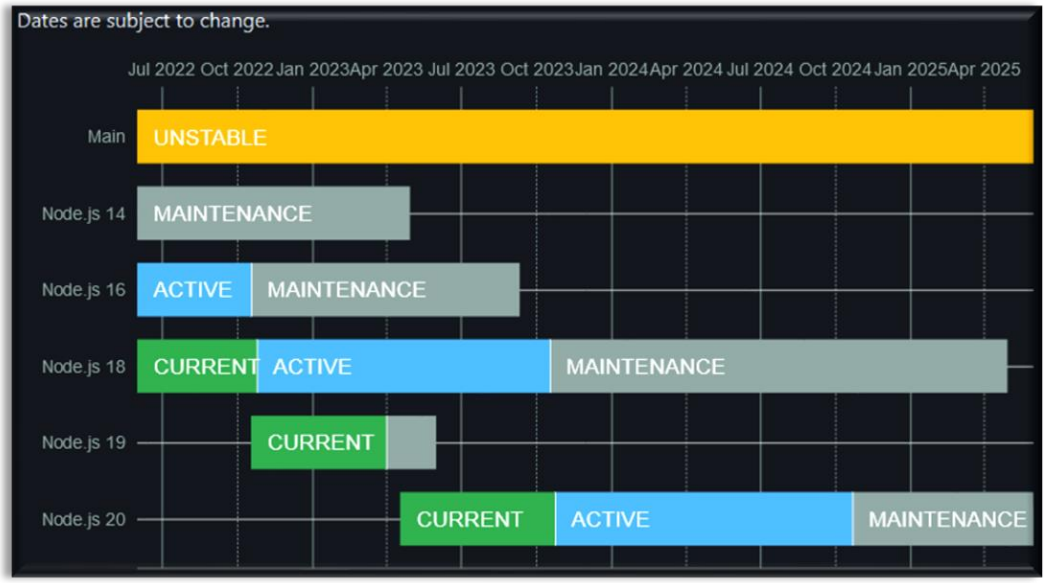
https://dzone.com/articles/how-good-is-nodejs-for-backend-development

Es un entorno controlado por eventos diseñado para crear aplicaciones escalables, permitiéndo establecer y gestionar múltiples conexiones al mismo tiempo. Gracias a esta característica, no tenemos que preocuparnos con el bloqueo de procesos, ya que no hay bloqueos.



Lo primero que tenemos que hacer, es instalar Node.js en nuestra máquina. Ya sea con Windows, Linux o Mac, el sitio web de Node.js (https://nodejs.org), detectará nuestro sistema operativo, y nos ofrecerá un paquete con el que instalarlo





Simplemente se pulsa en el botón verde de la versión que se requiera (la recomendación es siempre usar las **versiones LTS**) y completar el proceso de instalación con las opciones por defecto. Una vez se ha instalado, para comprobar que todo funciona correctamente, se abre una terminal (en windows, CMD o PowerShell valen perfectamente) y escribimos los siguientes comandos:

`node -v`	Este comando nos devolverá la versión de Node.js que se ha instalado
`npm -v`	Este comando nos devolverá la versión de NPM que hay instalada
`npm install npm -g`	Este comando permite actualizar la versión de NPM

Con esto, ya tenemos instalado **Node.js** y **NPM** (Gestor de paquetes) que es todo lo que necesitamos inicialmente para comenzar

https://blog.siddu.tech/new-in-nodejs-v18

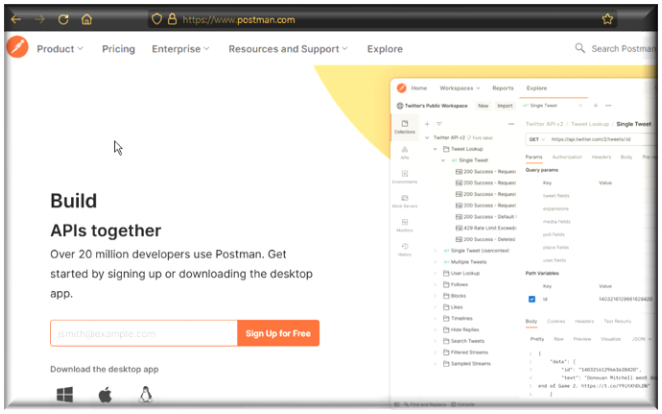
npm (Node Package Manager)	
https://www.npmjs.com/	

npm es un gestor de paquetes desarrollado en su totalidad bajo el lenguaje JavaScript por Isaac Schlueter, a través del cual podemos obtener cualquier librería con tan solo una sencilla línea de código, lo cual nos permitirá agregar dependencias de forma simple y distribuir paquetes.

Así mismo, es de gran importancia mencionar que Node.JS desde su versión 0.6.3 se instala automáticamente desde el entorno NPM, permitiendo a los desarrolladores instalar aplicaciones Node que se encuentren en el repositorio.

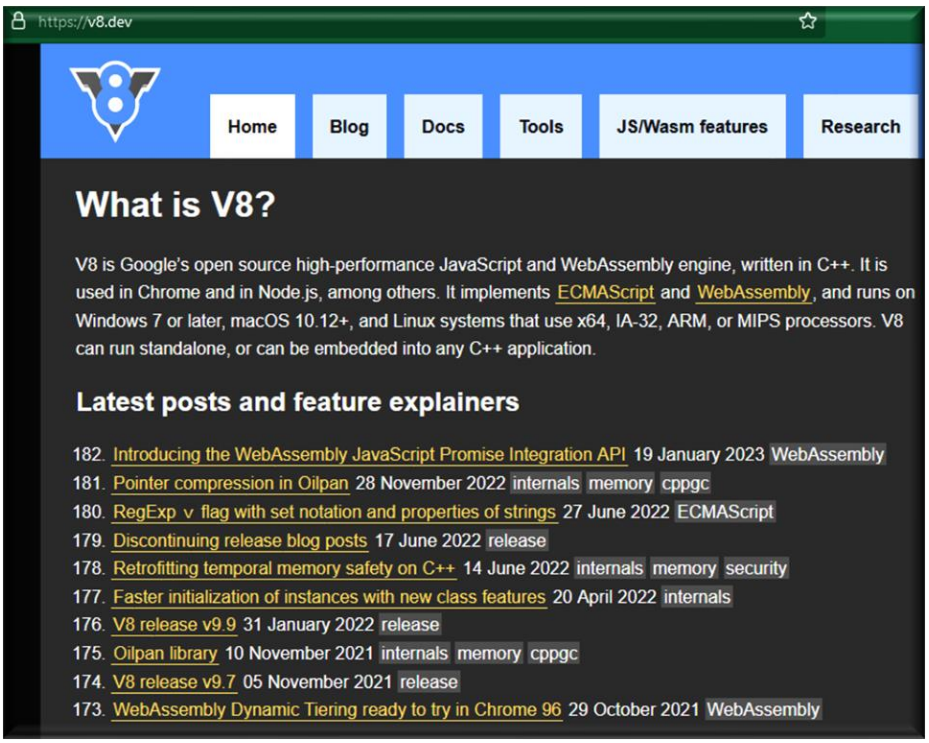
Postman	
https://www.postman.com	

Postman es una aplicación que nos permite realizar pruebas API. Es un cliente HTTP que nos da la posibilidad de testear 'HTTP requests' a través de una interfaz gráfica de usuario, por medio de la cual obtendremos diferentes tipos de respuesta que posteriormente deberán ser validados



Motor V8	
https://v8.dev/	

Es un motor open-source escrito en C++ para compilar JavaScript y WebAssembly en código máquina. Es un entorno de ejecución de Js, creado por Google y liberado en el 2008. Está escrito en c++ y convierte el Javascript en código máquina en lugar de interpretarlo



ECMAScript

https://tc39.es/ecma262/

https://www.ecma-international.org/

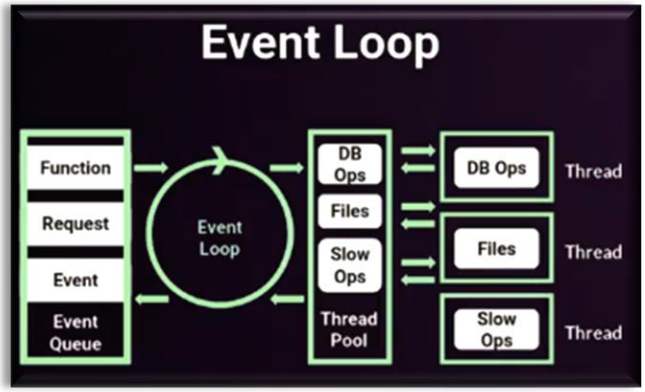
Características principales de JavaScript:

- **Concurrencia**: Es monohilo, con entradas y salidas asíncronas.
- **Motor V8**: Creado por Google en 2008 para Chrome. Escrito en C++. Convierte JS en código máquina en lugar de interpretarlo en tiempo real.
- Todo funciona en base a **Módulos**, que son piezas de código muy pequeñas que modularizan nuestros sistemas y ayudan a entender mejor el código.
- **Orientación a Eventos**, existe un bucle de eventos que se ejecuta constantemente. Lo que nos permite programar de forma reactiva, lo que quiere decir que podemos programar con la lógica de "Cuando sucede algo, se ejecuta esta parte de mi código y eso a su vez dispara otra parte"

Event Queue: Contiene todos los eventos que se generan por nuestro código (Funciones, peticiones, etc.), estos eventos quedan en una cola que van pasando uno a uno al Event Loop.

Event Loop: Se encarga de resolver los eventos ultra rápidos que llegan desde el Event Queue. En caso de no poder resolverse rápido, enviá el evento al Thread Pool.

Thread Pool: Se encarga de gestionar los eventos de forma asíncrona. Una vez terminado lo devuelve al Event Loop. El Event Loop vera si lo pasa a Event Queue o no.



PROCESO DE NODE

1.- Va a abrirse un proceso, ese proceso es un proceso de node
2.- Interpreta todo el archivo
3.- Convertirlo a código maquina
4.- Prepara todo lo que necesita para ejecutarse
5.- Se ejecuta
6.- Se cierra el proceso, y termina

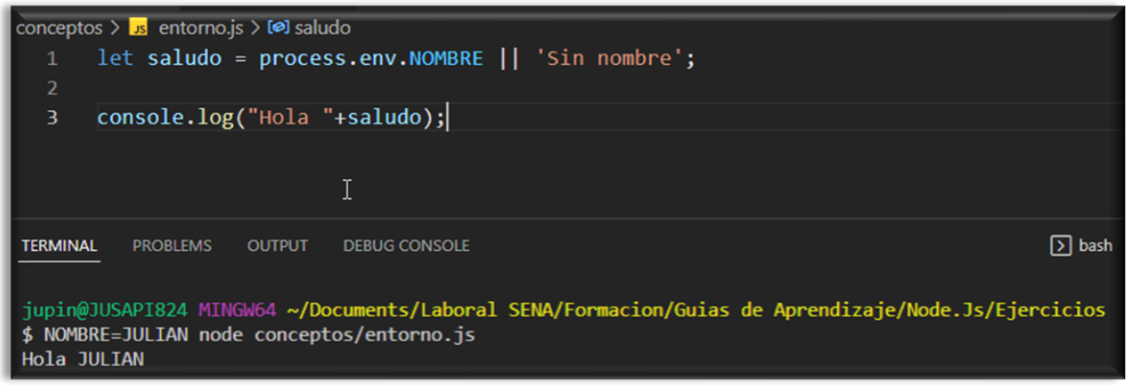
```
conceptos > monohilo.js > ...
1   console.log('Hola Mundo!');
2   let i=0;
3   setInterval(function(){
4       console.log('sigo activo');
5       if(i===5){
6           i = i + z;
7       }
8       i++;
9   }, 1000);
10
```

```
jupin@JUSAPI824 MINGW64 ~/Documents/Laboral SENA/Formacion/Guias de Aprendizaje/Node.Js/Ejercicios
$ node conceptos/monohilo.js
Hola Mundo!
sigo activo
sigo activo
sigo activo
sigo activo
sigo activo
sigo activo
C:\Users\jupin\Documents\Laboral SENA\Formacion\Guias de Aprendizaje\Node.Js\Ejercicios\conceptos\monohilo.js:6
        i = i + z;
                ^

ReferenceError: z is not defined
    at Timeout._onTimeout (C:\Users\jupin\Documents\Laboral SENA\Formacion\Guias de Aprendizaje\Node.Js\Ejercicios\
conceptos\monohilo.js:6:17)
    at listOnTimeout (node:internal/timers:559:17)
    at processTimers (node:internal/timers:502:7)
```

Variables de entorno

Las variables de entorno son una forma de llamar información de afuera a nuestro software, sirve para definir parámetros sencillos de configuración de los programas de modo que puedan ejecutarse en diferentes ambientes sin necesidad de modificar el código fuente de un script.

El objeto **process** nos da información sobre el proceso que está ejecutando este script. La propiedad **env** es la que nos da acceso a las variables de entorno de manera sencilla**.**



https://jairofernandez.medium.com/manejo-de-variables-de-entorno-en-node-js-ac90f7a2c1e5

https://protegermipc.net/2018/11/22/permitir-la-ejecucion-de-scripts-powershell-en-windows-10/

Módulos en Node.js

Un módulo no es nada más que una unidad de código organizado en archivos o directorios, la cual puede ser exportada con facilidad para poder reutilizarse en otras partes de la aplicación.

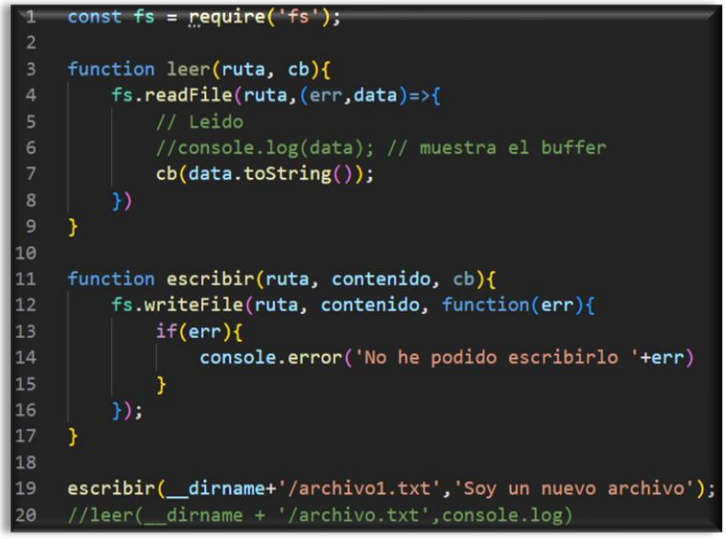
Un **módulo** puede contener una clase o una biblioteca de funciones para un propósito específico. Durante mucho tiempo, **JavaScript** existió sin una sintaxis de **módulo** a nivel de lenguaje. Eso no fue un problema, porque inicialmente los scripts eran pequeños y simples, por lo que no era necesario

Tipos de módulos

Hay 3 tipos de módulos. Todos funcionan de una manera similar pero difieren en el origen.

- **Built-in modules**: Son los módulos nativos de la API de Node.js. No hace falta que se instalen, ya que vienen incluidos por defecto con Node.js. Algunos ejemplos son los módulos **fs** o **stream**. Estos paquetes solo son actualizados si cambias la versión de Node.js.

- **Local modules:** Son los módulos escritos por los desarrolladores y forman en su conjunto gran parte de la aplicación. Se estructuran así con la finalidad de poder ser un código reutilizable.
- **External modules:** Son, en esencia, los paquetes de terceros distribuidos a través de npm (aunque pueden provenir de otros repositorios). Estos paquetes se instalan como dependencias y, aunque aportan funcionalidad a la aplicación, no deben incluirse en el repositorio ya que no son parte de la misma.



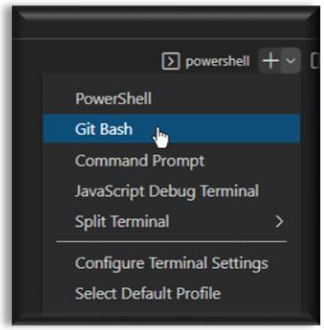
```
const fs = require('fs');

function leer(ruta, cb){
    fs.readFile(ruta,(err,data)=>{
        // Leido
        //console.log(data); // muestra el buffer
        cb(data.toString());
    })
}

function escribir(ruta, contenido, cb){
    fs.writeFile(ruta, contenido, function(err){
        if(err){
            console.error('No he podido escribirlo '+err)
        }
    });
}

escribir(__dirname+'/archivo1.txt','Soy un nuevo archivo');
//leer(__dirname + '/archivo.txt',console.log)
```

- **Nodemon (Desarrollo):** Es un gestor que nos ayuda a detectar los cambios, compilarlo y ejecutarlos.
    - `npm install -g nodemon`

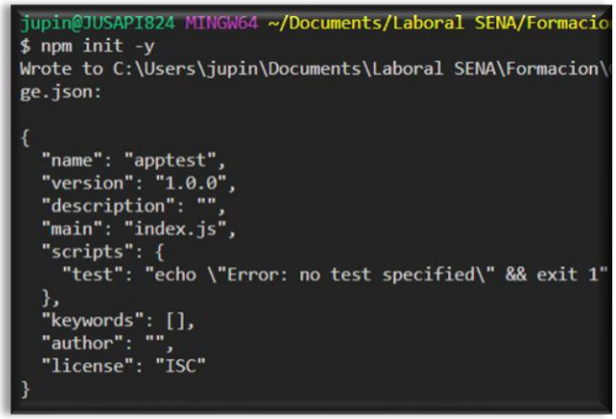
- **PM2 (Producción):** Es parecida a nodemon simplemente mas avanzada y mas compleja. No se debe utilizar en desarrollo por que dará más problemas que soluciones. Nos ayudara a ver los datos de nuestra aplicación en producción, como el uso del CPU, memoria, cuantas veces se ha reiniciado.
    - `npm install -g pm2`

Creación primera aplicación de prueba

Para iniciar nuestra primer aplicación, primero debemos crear la carpeta y abrir VSCode. Abrimos una terminal de **GitBash**



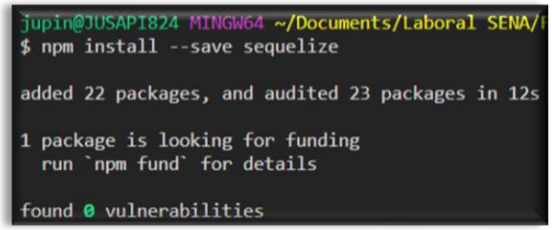
creamos el archivo **package.json** de la siguiente manera, o bien ingresando los datos con el comando `npm init - y`:



Instalación del ORM Sequelize

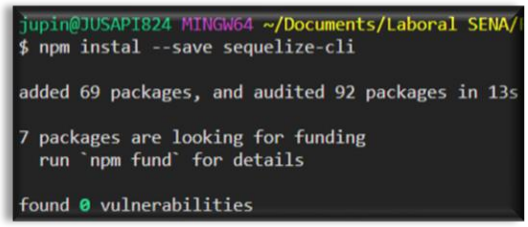
ORM (Object-Relational Mapping) es una técnica para convertir datos entre el sistema de tipos del lenguaje de programación y la base de datos. Como su nombre lo indica, esto va dirigido solamente a las base de datos relacional (SQL). **Sequelize es** un ORM para Nodejs que nos permite manipular varias bases de datos SQL de una manera bastante sencilla, entre estas bases de datos podemos encontrar: mysql, sqlite, postgres, mssql

Ejecutamos el comando: `npm install --save sequelize`



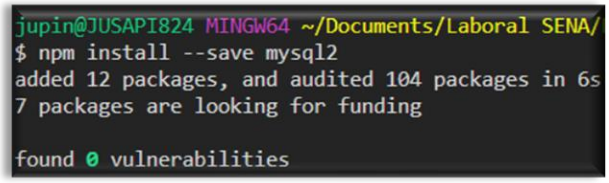
Con el fin de poder crear las migraciones y modelos desde la consola, ejecutamos el siguiente comando

```
npm install --save sequelize-cli
```



Nos conectaremos a una base de datos de mysql, por ello instalaremos el siguiente módulo:

npm install --save mysql2



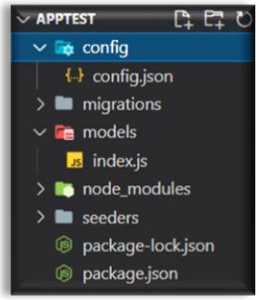
Después de instalar estos paquetes iniciales, ejecutamos un comando de NPX (**Node Package Execute)** que viene con **npm**, cuando se instalan versiones por encima de la 5.2.0. Es un corredor de paquetes npm que puede ejecutar cualquier paquete que desee del registro npm sin siquiera instalar ese paquete. El npx es útil durante un paquete de uso único. Se puede ejecutar `npx -v` para saber la versión.

Ejecutamos el siguiente comando para ver las opciones que tenemos con **sequelize:**

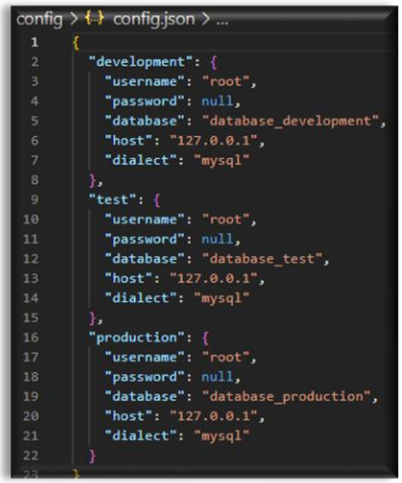
```
npx sequelize -help
```

```
jupin@JUSAPI824 MINGW64 ~/Documents/Laboral SENA/Formacion/Guias de Aprendizaje/Node.Js/
$ npx sequelize -help

Sequelize CLI [Node: 16.14.2, CLI: 6.4.1, ORM: 6.19.0]

sequelize <command>

Commands:
  sequelize db:migrate                        Run pending migrations
  sequelize db:migrate:schema:timestamps:add  Update migration table to have timestamps
  sequelize db:migrate:status                 List the status of all migrations
  sequelize db:migrate:undo                   Reverts a migration
  sequelize db:migrate:undo:all               Revert all migrations ran
  sequelize db:seed                           Run specified seeder
  sequelize db:seed:undo                      Deletes data from the database
  sequelize db:seed:all                       Run every seeder
  sequelize db:seed:undo:all                  Deletes data from the database
  sequelize db:create                         Create database specified by configuration
  sequelize db:drop                           Drop database specified by configuration
  sequelize init                              Initializes project
  sequelize init:config                       Initializes configuration
  sequelize init:migrations                   Initializes migrations
  sequelize init:models                       Initializes models
  sequelize init:seeders                      Initializes seeders
```

Allí encontramos todas las opciones que tenemos con el ORM y ejecutamos el comando: **`npx sequelize init`** para inicializar el proyecto. De esta manera ya nuestro proyecto tiene los siguientes carpetas (archivo de configuración de la base de datos en diferentes ambientes, carpeta de Modelos, Migraciones y Seeders):

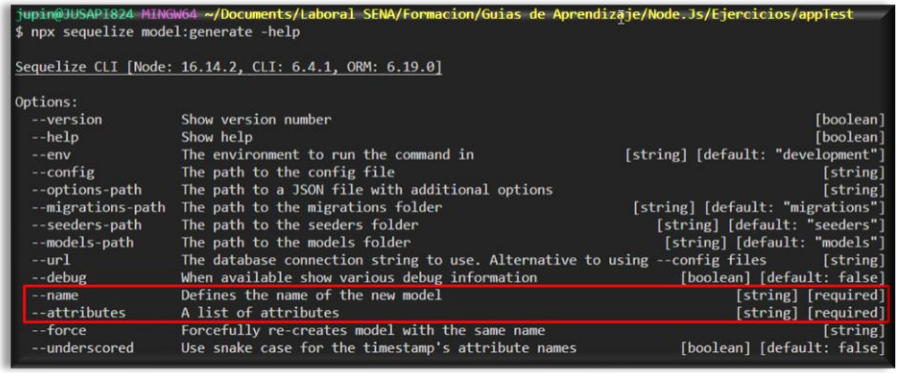


Archivo **config.json:**



Creación del modelo:

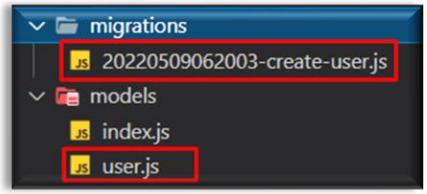
Podemos ejecutar el comando `npx sequelize model:generate -help` para identificar la forma en que se pueden crear los modelos



Crearemos el modelo **User** con los atributos name, email y password a través del siguiente comando:

npx sequelize model:generate --name User --attributes name:STRING,email:STRING,password:STRING

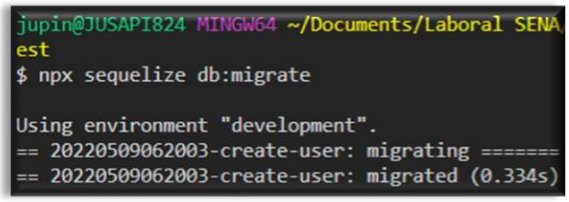
Después de estos comando se nos crean dos archivos **Modelo**(donde luego podremos definir asociaciones y relaciones) y la **Migración** (Tiene los médotos **up**(Crear la tabla) y **down**(Crear la tabla))



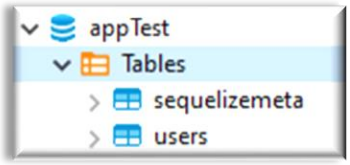
Migrar la base de datos a nuestra base de datos MySQL

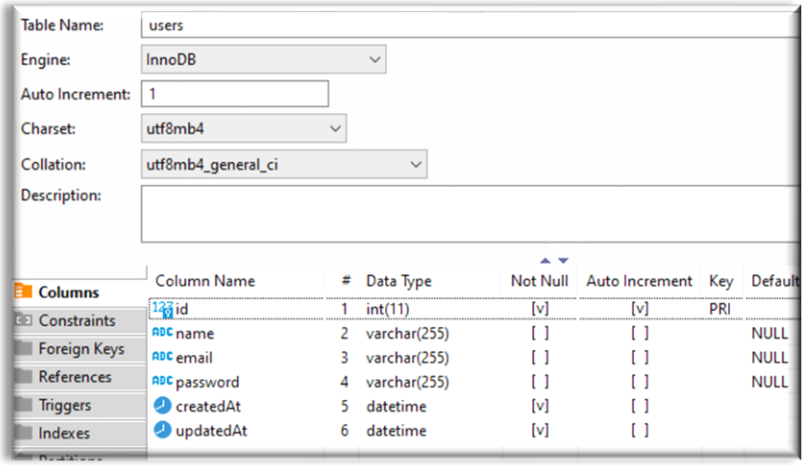
Ahora vamos a ejecutar la migración que acabamos de crear para que se pueda crear la tabla en nuestra base de datos creada:

```
npx sequelize db:migrate
```



Después de ejecutar el comando, Sequelize nos crea las siguientes tablas:





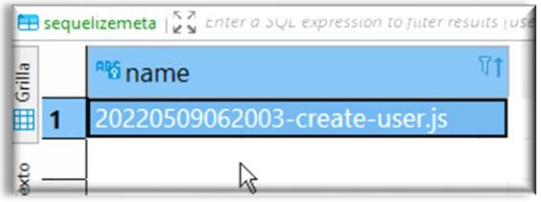
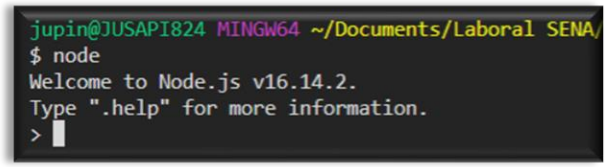


Tabla creada para llevar un control de las migraciones

Inicialmente ejecutaremos todo desde la consola de comandos donde veremos insertar un registro, para ello escribimos:

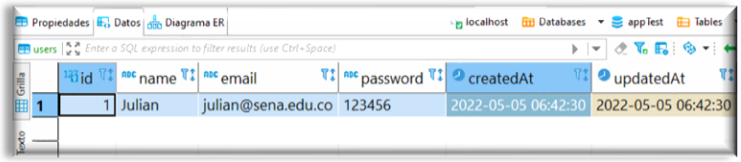
node



Una vez que node está corriendo ejecutamos lo siguiente:

```
jupin@JUSAPI824 MINGW64 ~/Documents/Laboral SENA/Formacion/Guias de Aprendizaje/Node.Js/Ejercicios/appTest
$ node
Welcome to Node.js v16.14.2.
Type ".help" for more information.
> const db = require('./models');
undefined
> const user = db.User.build({name:"Julian",email:"julian@sena.edu.co",password:"123456" });
undefined
> user.save().then(console.log).cath(console.error);
Uncaught TypeError: user.save(...).then(...).cath is not a function
> Executing (default): INSERT INTO `Users` (`id`,`name`,`email`,`password`,`createdAt`,`updatedAt`) VALUES (D
EFAULT,?,?,?,?,?);
User {
  dataValues: {
    name: 'Julian',
    email: 'julian@sena.edu.co',
    password: '123456',
    updatedAt: 2022-05-09T06:42:30.878Z,
    createdAt: 2022-05-09T06:42:30.878Z
  },
  _previousDataValues: {
    name: 'Julian',
    email: 'julian@sena.edu.co',
    password: '123456',
    id: 1,
    createdAt: 2022-05-09T06:42:30.878Z,
    updatedAt: 2022-05-09T06:42:30.878Z
  },
  uniqno: 1,
  _changed: Set(0) {},
  _options: { isNewRecord: true, _schema: null, _schemaDelimiter: '' },
  isNewRecord: false
}
```

Podemos corroborar que en nuestra tabla de usuarios ya se encuentra el nuevo registro



Retornar todos los usuarios de la tabla

```
jupin@JUSAPI824 MINGW64 ~/Documents/Laboral SENA/Formacion/Guias de Aprendizaje/Node.Js/Ejercicio
$ node
Welcome to Node.js v16.14.2.
Type ".help" for more information.
> const db = require('./models');
undefined
> db.User.find
undefined
> db.User.find
db.User.findAll          db.User.findAndCountAll  db.User.findByPk         db.User.findCreateFind
db.User.findOne          db.User.findOrBuild      db.User.findOrCreate
```

```
> db.User.findAll().then(console.log).cath(console.error);
Uncaught TypeError: db.User.findAll(...).then(...).cath is not a function
> Executing (default): SELECT `id`, `name`, `email`, `password`, `createdAt`, `updatedAt` FROM `Users` AS `Us
er`;
[
  User {
    dataValues: {
      id: 1,
      name: 'Julian',
      email: 'julian@sena.edu.co',
      password: '123456',
      createdAt: 2022-05-09T06:42:30.000Z,
      updatedAt: 2022-05-09T06:42:30.000Z
    },
    uniqno: 1,
    _changed: Set(0) {},
    _options: {
      isNewRecord: false,
      _schema: null,
      _schemaDelimiter: '',
      raw: true,
      attributes: [Array]
    },
    isNewRecord: false
  }
]
>
```

Retornar un usuario específico

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE                                                    bash

undefined
> db.User.findByPk(1).then(console.log).catch(console.error);
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 191,
  [Symbol(trigger_async_id_symbol)]: 190,
  [Symbol(destroyed)]: { destroyed: false }
}
> Executing (default): SELECT `id`, `name`, `email`, `password`, `createdAt`, `updatedAt` FROM `Users` AS `Us
er` WHERE `User`.`id` = 1;
User {
  dataValues: {
    id: 1,
    name: 'Julian',
    email: 'julian@sena.edu.co',
    password: '123456',
    createdAt: 2022-05-09T06:42:30.000Z,
    updatedAt: 2022-05-09T06:42:30.000Z
  },
  _previousDataValues: {
    id: 1,
    name: 'Julian',
    email: 'julian@sena.edu.co',
    password: '123456',
    createdAt: 2022-05-09T06:42:30.000Z,
    updatedAt: 2022-05-09T06:42:30.000Z
  },
```

Actualizar un usuario

```
> db.User.update({password: '654321'},{where: {id:1}}).then(console.log).catch(console.error);
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 1077,
  [Symbol(trigger_async_id_symbol)]: 1076,
  [Symbol(destroyed)]: { destroyed: false }
}
> Executing (default): UPDATE `Users` SET `password`=?,`updatedAt`=? WHERE `id` = ?
[ 1 ]
```

- **Actividades de transferencia del conocimiento**
  Los aprendices deberán demostrar los conocimientos conceptuales y procedimentales que adquirieron a partir de las actividades de apropiación mediante el siguiente instrumento de evaluación:

Los aprendices deberán realizar cada una de las actividades propuestas para adquirir dominio en los conocimientos relacionados con los diferentes temas. Se evaluarán los conocimientos adquiridos mediante instrumentos de evaluación de Desempeño y Producto relacionados en la presente guía

- **Ambiente Requerido**
  Ambiente de SISTEMAS con conexión eléctrica e internet
- **Materiales**
  - Computadores (30)
  - Sillas (3)
  - Televisor (1)
  - Resma tamaño carta (1)
  - Marcadores (3)
  - Lápiz (1)
  - Lapicero (1)

4. ACTIVIDADES DE EVALUACIÓN

Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación
Evidencias de Conocimiento: **Evidencias de Desempeño:** Asistencia y participación activa en las diferentes actividades propuestas **Evidencias de Producto:** Respuestas y procedimiento de los talleres realizados	Crea la base de datos en el motor de base de datos seleccionado, siguiendo especificaciones técnicas del informe, según normas y protocolos de la empresa.	**Observación:** EXC_D_01 **Valoración del Producto:** EXC_P_01

5. GLOSARIO DE TÉRMINOS

Sistema de información: es un conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior, generados para cubrir una necesidad o un objetivo

Sistema operativo – Es un conjunto de programas que sirven para manejar un ordenador.

Software - El conjunto de programas, procedimientos y documentación asociado a un sistema informático.

Javascript: es un lenguaje de programación del lado del cliente que se utiliza con frecuencia en diseño WEB para generar efectos más complejos que no se puedan alcanzar usando HTML.

HTML: Siglas de las palabras inglesas: Hypertext Markup Language. Es decir, lenguaje de marcado de hipertexto. Lenguaje informático para crear páginas web. Conjunto de etiquetas o instrucciones que permiten estructurar el contenido de una web e incluir los hipervínculos o enlaces enlaces a otras páginas. Este lenguaje lo inventó en 1991 el Doctor Berners-Lee del CERN en Suiza.

HTTPS: Siglas de las palabras inglesas: HyperText Transfer Protocol Secure o versión segura del protocolo HTTP. Es el protocolo empleado para la transferencia de ficheros HTML cifrados que puedan contener información confidencial.

HTTP: siglas de las palabras inglesas: Hypertext Transfer Protocol. A saber en español: Protocolo de Transmisión de Hipertexto. Protocolo estándar de transferencia de hipertexto. Es decir: el protocolo de comunicaciones en el que está basado la Word Wide Web.

Script: es un archivo de órdenes o archivo de procesamiento por lotes. Es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.

MySQL: es un sistema de gestión de bases de datos de código abierto que, junto con PHP, permite darle a las páginas web cierto dinamismo, es decir, disponer de manera adecuada los datos solicitados por los navegadores. Es un sistema multiplataforma y su uso está tan extendido en las bases de datos que podría considerarse un estandar.

SEO (Search Engine Optimisation) Optimización en buscadores: técnica utilizada para asegurar que una página Web es compatible con los motores de búsqueda y así tener la posibilidad de aparecer en las posiciones más altas en los resultados de búsqueda.

Diseño web adaptable (responsive web design): se llama así al diseño web de aquellas páginas que se adaptan al tamaño de la pantalla o ventana en que se despliegan, por medio del uso de, idealmente, un solo documento HTML y un solo documento CSS. Esto permite hacer una sola página web para smartphones, phablets, tablets y PC.

Diagrama o Modelo Entidad Relación (DER): denominado por sus siglas en inglés, E-R "Entity relationship", o del español DER "Diagrama de Entidad Relación") es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades

Bases de Datos (BD): es un banco de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto.

6. REFERENTES BILBIOGRÁFICOS

- Documentos técnicos relacionados en la plataforma
- https://www.youtube.com/watch?v=zV6W9GrWLPg
- https://www.youtube.com/watch?v=u2Ms34GE14U

7. CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	Julian Salazar Pineda	Instructor	Centro de Procesos Industriales y Construcción	14 Abril de 2022

8. CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)	Julian Salazar Pineda	Instructor	Centro de Procesos Industriales y Construcción	Febrero 2023	Actualización imágenes e información guía