UNIVERSIDAD CATÓLICA DE HONDURAS

"NUESTRA SEÑORA REINA DE LA PAZ"

Alumno:

Oscar Andree Varela Godoy

Catedrático:

Carlos Antonio Flores

Asignatura:

Programación Móvil II

Tema:

Investigación #1

Indice

Contenido

Introducción:

Por medio de esta investigación (Glosario) buscamos encontrar mayor información de los paquetes para poder comprender el funcionamiento de cada uno de ellos, empezando desde conocer que son e investigando sobre su correcto funcionamiento para el momento de tener que utilizarlo ya estar familiarizado un poco con cada paquete que se investigo

Objetivos:

Objetivos Generales:

- Conocer que son los paquetes
- Saber cómo funcionan

Objetivos específicos:

- Reconocer que hace cada uno de los paquetes que se investigaron para dar un mejor uso a cada uno de estos.

Contenido:

Morgan:

Morgan es un Middleware de nivel de solicitud HTTP. Es una gran herramienta que registra las requests junto con alguna otra información dependiendo de su configuración y el valor predeterminado utilizado. Demuestra ser muy útil durante la depuración y también si desea crear archivos de registro.

Como se usa:

- Paso 1: cree una nueva carpeta para un proyecto con el siguiente comando:
  - mkdir morgan
- Paso 2: Navega a nuestra carpeta usando el siguiente comando:
  - cd morgan
- Paso 3: inicialice npm usando el siguiente comando y archivo de servidor:
  - npm init -y
  - toque index.js
- Paso 4: instale los paquetes necesarios con el siguiente comando:
  - npm expreso Morgan

nodemon:

Nodemon es una de las muchas herramientas que nos proporciona npm para mejorar o añadir algunas funcionalidades extra y normalmente muy útil útiles a Node.js

¿Para qué Sirve Nodemon?
Nodemon se ha creado para facilitar el desarrollo en Node.js añadiendo una nueva función que nos va a venir muy bien, que es la de permitir que todos los cambios que realicemos en nuestro proyecto se implementen a tiempo real en este.

Como se usa:
Su uso es muy sencillo, simplemente hay que utilizar el comando nodemon en lugar de node al ejecutar un archivo .js:

nodemon app.js

Al hacerlo la terminal crea un proceso, esto mantendrá a la terminal ocupada.

Para salir del proceso simplemente se pulsa CTRL + C y se puede utilizar nuevamente la terminal.

Express:

Básicamente es un marco de desarrollo minimalista para Node.js que permite estructurar una aplicación de una manera ágil, nos proporciona funcionalidades como el enrutamiento, opciones para gestionar sesiones y cookies, etc.

Instalación y uso:

Lo primero que debemos hacer es crear un directorio que usaremos para contener nuestra aplicación y así convertirlo en nuestro directorio de trabajo.

- mkdir myapp
- cd myapp

El siguiente paso es usar npm para iniciar la creación de un archivo json para nuestra aplicación.

- npm init

Este comando nos solicita información previa, como el nombre y la versión de su aplicación. Pero por el momento y para probar, bastaría con presionar INTRO para aceptar los valores predeterminados para la mayoría de ellos, con la siguiente excepción:

- entry point: (index.js)

Aquí definimos app.js, o el nombre que queramos para el archivo principal. También puedes aceptar el valor por defecto y que sea index.js presionando INTRO para aceptar el nombre de archivo predeterminado sugerido.
El siguiente paso es instalar Express en el directorio myapp y guardarlo en la lista de dependencias. Por ejemplo:

- npm install express --save

Bcrypt:

Bcrypt es una función de hashing de passwords diseñado por Niels Provos y David Maxieres, basado en el cifrado de Blowfish. Se usa por defecto en sistemas OpenBSD y algunas distribuciones Linux y SUSE. Lleva incorporado un valor llamado salt, que es un fragmento aleatorio que se usará para generar el hash asociado a la password, y se guardará junto con ella en la base de datos. Así se evita que dos passwords iguales generen el mismo hash y los problemas que ello conlleva, por ejemplo, ataque por fuerza bruta a todas las passwords del sistema a la vez.

Como se usa:.

```
var bodyParser = require('body-parser');
var bcrypt = require('bcrypt');
var usersDB = require('usersDB');

app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))

var BCRYPT_SALT_ROUNDS = 12;
app.post('/register', function (req, res, next) {
  var username = req.body.username;
  var password = req.body.password;

  bcrypt.hash(password, BCRYPT_SALT_ROUNDS)
    .then(function(hashedPassword) {
        return usersDB.saveUser(username, hashedPassword);
    })
    .then(function() {
        res.send();
    })
    .catch(function(error){
        console.log("Error saving user: ");
        console.log(error);
        next();
    });
});
```

```
app.post('/login', function (req, res, next) {
  var username = req.body.username;
  var password = req.body.password;

  usersDB.getUserByUsername(username)
    .then(function(user) {
        return bcrypt.compare(password, user.password);
    })
    .then(function(samePassword) {
        if(!samePassword) {
            res.status(403).send();
        }
        res.send();
    })
    .catch(function(error){
        console.log("Error authenticating user: ");
        console.log(error);
        next();
    });
});
```

Express-Validator:

express-validator es un conjunto de middlewares express.js que envuelve las funciones de validación y desinfección de validator.js.

Como se usa:

- Cree la carpeta del proyecto de validación de entrada rápida ejecutando el siguiente comando.
  - mkdir express-node-form-validation
- Entra en el directorio del proyecto.
  - cd express-node-form-validation
- Ejecutar comando para crear package.json:
  - npm init

Sequelize:

Sequelize es un ORM de Node.js basado en promesas para Postgres, MySQL, MariaDB, SQLite y Microsoft SQL Server. Sus características son soporte de transacciones sólido, relaciones, carga ansiosa y perezosa, replicación de lectura y muchas más.

Configurando una aplicación Node.js:

- Inicie la aplicación Node.js con el siguiente comando:
  - npm init -y

Instalación de Sequelize:

- Sequelize necesita el módulo MySql instalado en su proyecto. Si no ha instalado el módulo MySql, asegúrese de que antes de instalar Sequelize necesita instalar el módulo MySql2 . Necesita instalar este módulo usando el siguiente comando.
  - npm instalar mysql2
- Después de instalar el módulo MySql2 , tenemos que instalar el módulo Sequelize para instalar este módulo usando el siguiente comando.
  - npm install sequelize

Módulo que requiere:

- Debe incluir el módulo Sequelize en su proyecto utilizando estas líneas.
  - const Sequelize = require ('sequelize')

Configurando el archivo database.js :

- // Include Sequelize module
- const Sequelize = require('sequelize')

- // Creating new Object of Sequelize
- const sequelize = new Sequelize(
- 'DATABASE_NAME',
- 'DATABASE_USER_NAME',
- 'DATABASE_PASSWORD', {

- // Explicitly specifying
- // mysql database
- dialect: 'mysql',

- // By default host is 'localhost'
- host: 'localhost'
- }
-);

- // Exporting the sequelize object.
- // We can use it in another file
- // for creating models
- module.exports = sequelize

MYSQL2:

Cliente MySQL para Node.js con enfoque en rendimiento. Admite declaraciones preparadas, codificaciones no utf8, protocolo de registro binario, compresión, ssl.

MySQL2 es principalmente API compatible con mysqljs y admite la mayoría de las funciones. MySQL2 también ofrece estas características adicionales

- Más rápido / mejor rendimiento
- Declaraciones preparadas
- Protocolo de registro binario de MySQL
- Servidor MySQL
- Soporte extendido para Codificación y Cotejo
- Envoltorio de promesa
- Compresión
- SSL y conmutador de autenticación
- Flujos personalizados
- puesta en común

Instalacion:

MySQL2 está libre de enlaces nativos y se puede instalar en Linux, Mac OS o Windows sin ningún problema.

```
npm instalar --guardar mysql2
```

Uso:

```
// obtener el cliente
const  mysql  =  require ( 'mysql2' ) ;

// crea la conexión a la base de datos
const  connection  =  mysql . createConnection ( {
  host : 'localhost' ,
  usuario : 'root' , base de
  datos : 'test'
} ) ;

// ejecutar llamará internamente a prepare y query
connection . ejecutar (
  'SELECT * FROM `table` WHERE `name` = ? AND `age` > ?' ,
  [ 'Rick C-137' ,  53 ] ,
  function ( err ,  resultados ,  campos )  {
    console . log ( resultados ) ;  // los resultados contienen filas devueltas por la
```

```
    consola del servidor .log ( fields ) ;  // los campos contienen metadatos adicionales sobre los
resultados, si están disponibles

    // Si vuelve a ejecutar la misma declaración, se seleccionará de un caché LRU
    // lo que ahorrará tiempo de preparación de consultas y brindará un mejor rendimiento
  }
) ;
```

+

Bibliografía:

- *¿Qué es MORGAN en Node.js? – Acervo Lima*. (s. f.). Morgan. Recuperado 22 de mayo de 2022, de https://es.acervolima.com/que-es-morgan-en-node-js/#:%7E:text=Morgan%3A%20Morgan%20es%20un%20Middleware,y%20el%20valor%20predeterminado%20utilizado.
- S. (2020, 1 enero). *Qué es Nodemon y cómo Instalarlo*. Silver Sites. https://www.silversites.es/desarrollo-web/que-es-nodemon/
- *Qué es Express.JS y primeros pasos*. (2022, 3 marzo). IfgeekthenNTTdata. Recuperado 22 de mayo de 2022, de https://ifgeekthen.nttdata.com/es/que-es-expressjs-y-primeros-pasos#:%7E:text=B%C3%A1sicamente%20es%20un%20marco%20de,se%20parece%20mucho%20a%20Connect.
- Izertis. (s. f.). Encriptación de password en NodeJS y MongoDB: bcrypt. Recuperado 22 de mayo de 2022, de https://www.izertis.com/es/-/blog/encriptacion-de-password-en-nodejs-y-mongodb-bcrypt
- Global, Z. (s. f.). *ZG España Zentica - Tutorial de Express Validator con ejemplos de validación de entrada*. Zentica Global. Recuperado 22 de mayo de 2022, de https://www.zentica-global.com/es/zentica-blog/ver/tutorial-de-express-validator-con-ejemplos-de-validacion-de-entrada-6073abf9dad90
- *¿Cómo usar Sequelize en Node.js? – Acervo Lima*. (s. f.). sequelize. Recuperado 22 de mayo de 2022, de https://es.acervolima.com/como-usar-sequelize-en-node-js/
- *npm: mysql2*. (2021, 14 noviembre). Npm. Recuperado 22 de mayo de 2022, de https://www.npmjs.com/package/mysql2