ABM ANGULAR- MYSQL

Parte I V1.0.1

Profesora de Entorno Virtual Luisina Benitez

Creamos una carpeta en el escritorio.

Abrimos la carpeta en VSC.

Abrimos la terminal en CMD y ejecutamos el comando: **npm init –yes** donde me va crear un archivo de package.json.

Luego vamos a instalar los módulos que vamos a utilizar, para eso vamos a ejecutar el comando:

npm i express express-handlebars

npm i express express-session

npm i express mysql

npm i express-mysql-session

npm i express morgan

npm i express bcryptjs

npm i express passport

npm i express passport-local

npm i express timeago.js

npm i express connect-flash

npm i express express-validator

Especificaciones de los comandos ejecutados. Colaboración de Marcelo Javier Luna Dalla Lasta (G20).

npm i express-handlebars
* es un motor de plantillas similar al módulo ejs en node. js, pero más potente y fácil de usar. Garantiza plantillas mínimas y es un motor sin lógica que mantiene la vista y el código separados. Se puede usar con express como el módulo hbs, disponible a través de npm.

npm i express express-session
* un marco del lado del servidor HTTP que se utiliza para crear y administrar un middleware de sesión

npm i express mysql
*

npm i express-mysql-session
*
npm i express morgan
*es un nodo. js y Express middleware para registrar solicitudes y errores HTTP, y simplifica el

proceso. En Nodo. js y Express, el middleware es una función que tiene acceso a los métodos del ciclo de vida de solicitud y respuesta, y al método next() para continuar con la lógica en su servidor Express.

npm i express bcryptjs
* La función hash de bcrypt nos permite construir una plataforma de seguridad de contraseñas que escala con el poder de cómputo y siempre procesa cada contraseña con un salt .

npm i express passport
* es un middleware de autenticación compatible con Express para Node. js . El único propósito de Passport es autenticar las solicitudes, lo que hace a través de un conjunto extensible de complementos conocidos como estrategias.

npm i express passport-local
* Este módulo le permite autenticarse usando un nombre de usuario y contraseña en su Node. aplicaciones js. Al conectarse a Passport, la autenticación local se puede integrar de manera fácil y discreta en cualquier aplicación o marco que admita el middleware de estilo Connect, incluido Express.

npm i express timeago.js
* es una biblioteca simple (solo 1kb) que se usa para formatear fecha y hora con la declaración `*** hace tiempo`. por ejemplo: 'hace 3 horas'

npm i express connect-flash
* permite a los desarrolladores enviar un mensaje cada vez que un usuario se redirige a una página web específica. Por ejemplo, cada vez que un usuario inicia sesión con éxito en su cuenta, se muestra (se muestra) un mensaje que indica su éxito en la autenticación.

npm i express express-validator
* es un conjunto de express. js middlewares que envuelve el validador. js funciones de validación y desinfección.

npm i nodemon -D
* es una herramienta que ayuda a desarrollar Node. js mediante el reinicio automático de la aplicación de nodo cuando se detectan cambios de archivo en el directorio. nodemon no requiere ningún cambio adicional en su código o método de desarrollo. nodemon es un contenedor de reemplazo para node.

Luego creamos una carpeta que se llame **src.**

Vamos a instalar otro módulo con el comando: npm i nodemon -D

Limpiamos la consola y nos ubicamos en la carpeta **src: cd src**

Vamos a crear las distintas carpetas dentro de la carpeta src: **mkdir lib public routers views**

Luego vamos hacer clic derecho sobre la carpeta y vamos a crear los archivos:

index.js

database.js

keys.js

Abrimos el archivo index.js y escribimos el código:

```
const express = require('express');
const morgan = require('morgan');
//inicializaciones: funciones de inicialización de clases
const app = express();
//configuraciones
app.set('port', process.env.Port || 4000);
//Middelwares: funciones que se ejecutan cada vez que un cliente solicita
una petición al servidor
app.use(morgan('dev'));
//Variables globales


//Routes

//Public: Código al cual va acceder el navegador


//satarting the server
app.listen(app.get('port'), () => {
    console.log('Serve on port', app.get('port'));

})
```

Luego vamos al archivo package.json y vamos a editar la parte de “script”:

```
"dev": "nodemon src/index.js"
```

Luego ejecutamos en la terminal: **npm run dev**

Vamos al navegador y en la URL colocamos: **localhost:4000** Nos da un error de servidor dado que no tiene ninguna ruta configurada.

Para configurar las rutas vamos a la carpeta “routes” y creamos el archivo index.js

En el archivo escribimos el siguiente código:

```
//Acá vamos a guardar todas las ruta principales de nuestro proyecto
const express = require('express');
const router = express.Router();
```

```
router.get('/', (req, res) =>{
    res.send("Hola Mundo");

});

module.exports = router;
```

Luego vamos al archivo index.js que está dentro de la carpeta src y en la sección //Routes escribimos:

```
//Routes
app.use(require('./routers/index.js'));
```

Luego verificamos en el navegador que haya aparecido el mensaje que hemos enviado por js.

En el index.js de la carpeta src vamos a importar las plantillas de express:

```
const { engine } = require('express-handlebars');
```

En la parte de //configuraciones escribimos:

```
app.set('port', process.env.Port || 4000);
app.path('views', path.join(__dirname, 'views'));
app.engine('.hbs', engine({
    defaultlayout: "main",
    layoutsDir: path.join(app.get('views'), 'layouts'),
    partialsDir: path.join(app.get('views'), 'partials'),
    extname: '.hbs'
}))
```

Dentro de la carpeta “views” creamos otra carpeta llamada “partials” donde vamos a guardar archivos con partes de códigos que vamos a utilizar.

Dentro de la carpeta “libs” vamos a crear un archivo que se llama “handlebars.js”

Dentro del archivo index.js que esta en la carpeta src agregamos en configuraciones:

```
helpers: require('./lib/handlebars.js')
```

y debajo de la función escribo:

```
app.set('view engine', '.hbs');
```

En la parte de “variables globales” agregamos:

```
app.use((req, res, next)=>{
    next();
})
```

Adentro de la carpeta “routers” creamos el archivo:

autentication.js

links.js

En el archivo index.js que esta adentro de src en la parte de “Routes” escribimos:

```
app.use(require('./routers/index.js'));
app.use(require('./routers/autentication'));
app.use('/links', require('./routers/links'));
```

Adentro de los archivos autentication.js y links.js escribimos lo siguiente:

```
const express = require('express');
const router = express.Router();

module.exports = router;
```

BASE DE DATOS

Creamos una carpeta en el raíz del proyecto que se llame: “database”

Adentro de database creamos un archivo que se llame: “db.sql” y escribimos los scripts para crear una base de datos y la creación de las tablas.