**UPTASK**

## NOTAS DEL PROYECTO:

Primer proyecto del curso de **Node.js** de **Udemy**:

<https://www.udemy.com/nodejs-bootcamp-desarrollo-web-mvc-y-rest-apis/learn/v4/content>

Primeros pasos:

* Crear carpeta con el nombre del proyecto.
* Consola: npm init
  + Crear el *package.json*
* Instalar dependencias:
  + npm install --save express
  + npm install --save-dev nodemon
* Crear *index.js*
  + Habilitar *express*:

const express = require(‘express’);

* + Configurar la aplicación:

const app = express();

* + Ruta para el inicio:

app.use(‘/’, (req, res) => {

res.send(‘Hola’);

});

* + Determinar puerto:

app.listen(3000);

* *Package.json*:
  + Formular “*scripts”*:

“start”: “nodemon ./index.js”;

* Crear carpeta + archivo *routes/index.js*
  + Ruta de enrutamiento en *index.js*

Const routes = require(‘./routes’);

app.use(‘/’, routes());;

* + *Routes/index.js*

const express = require(‘express’);

const router = express.Router();

module.exports = function() {

router.get(‘/’, (req, res) => {

res.send(‘Index’);

});

return router;

}

El sistema MODEL-VIEW-CONTROLLER (MVC):

* **Model**:

Encargado de los datos (desde una base de datos) y de la lógica destinada a mostrar esos datos.

*Ejemplo: un usuario quiere ver la sección de productos. El modelo se encarga de realizar la consulta pertinente a la base de datos con el fin de obtener los productos.*

* **View**:

Se encarga de lo que se imprime en la pantalla (html).

*Ejemplo: si el modelo hace la consulta a la base de datos para obtener los productos, la vista es la que se ocupa de mostrar los resultados obtenidos de la consulta a la BD.*

* **Controller**:

Intercede entre el modelo y la vista, antes de que el modelo realice la consulta a la base de datos, se encarga de llamar a la misma y, una vez obtenidos los resultados de la consulta a la base de datos, se ocupa de pasarlos o enviarlos a la vista.

* **Router**:

Encargado de soportar todas las URL’s o *endpoints* que gestiona la aplicación. Si el usuario accede a “*/productos*”, el Router llama a un controlador, que se encarga de comunicarse con el modelo (base de datos), para obtener los resultados de la consulta realizada y gestionar su envío a la vista que se ocupará de mostrar dicho resultado.

TEMPLATE ENGINE:

Son las vistas del MVC.

Se encargan de mostrar la parte visual (html) de en una aplicación Express, debido a que el modelo retorno un *objeto* (o arreglo) de datos; permite acceder a los resultados de una consulta y mostrarlos en pantalla

npm install --save pug

**ENLACE GIST TO MYSQL**

<https://gist.github.com/juanpablogdl/07bc2be083901a8e0b8c46844762394e>

<https://dev.mysql.com/downloads/file/?id=483310>

*Please enter a password for the root user:*

rootroot

**ENLACE TABLEPLUS**

<https://tableplus.io/>

Una vez instalada la herramienta, configuramos una nueva conexión de tipo *MySQL*.

**QUÉ ES UN ORM**

**Object Relational Mapping**

Permite almacenar/leer objetos de una base de datos.

* Object:

Se definen con un lenguaje de programación (en este caso con *javascript*). El archivo Model.

* Relational:

En definitiva, es lo que viene siendo la Base de Datos.

* Mapping:

La unión de los dos anteriores en la aplicación web.

Evita la repetición de código, ya que el modelo sólo se define en un único lugar y solo con importar el modelo al controlador se puede acceder a todos los métodos del ORM.

Evita escribir código MySQL.

Permiten ser escritos en el mismo lenguaje en el que se está desarrollando la aplicación.

ORM para NODE

1. *Sequelize*
2. *PostgreSQL*
3. *SQL Server*
4. *SQLite*

Mongoose es otro ORM que permite conectar MONGODB con las aplicaciones NODE.

Modelo y ORM

El *modelo* es un lugar único donde se describe la forma de los objetos y es más fácil agregar o quitar campos para la base de datos, mientras que el *ORM* tiene los métodos para la base de datos.



**INSTALAR SEQUELIZE**

npm install --save mysql2 sequelize

Se crea la carpeta y el archivo *Config>db.js*

|  |
| --- |
| ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'password'; |
|  | FLUSH PRIVILEGES; |

**INSTALAR VALIDATOR**

npm install --save validator

**INSTALAR SLUG**

Para las urls

npm install --save slug

**INSTALAR SHORTID**

Para las urls

npm install --save shortid

**INSTALAR BABEL – CONFIGURAR WEBPACK**

Para las urls

npm install -D babel-loader @babel/core @babel/preset-env webpack

**Install concurrently**

npm install --save concurrently