|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Carrera** | **Plan de estudios** | **Clave materia** | **Nombre de la materia** | **Semestre** | **Grupo** | **Periodo** |
| **ISC** | **ISIC-2010-** | DAM-1904 | Desarrollo de Aplicaciones para el  Comercio Electrónico | **9** |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Practica No.** | **Laboratorio de:** | **Nombre de la practica** | **Duración (Hora)** |
| **3** | **CÓMPUTO** | APLICACIONES PARA EL LADO DEL SERVIDOR | **10HRS** |

|  |  |
| --- | --- |
| **Unidad temática: 1** | **Subtema:**  CONCEPTOS BÁSICOS DE LA HERRAMIENTA DE DESARROLLO DEL LADO DEL SERVIDOR (login) |
| **Competencia especifica materia:**  Desarrolla aplicaciones para comercio electrónico en la nube utilizando software libre. | |
| **Competencia de la práctica:**  Crea y aplica funcionalidades especiales para aplicaciones de comercio electrónico  como proyecto final | |
| **Materiales:**  Visual Code, librerías API, Dependencias, REST | |
| **Desarrollo:**  Primero vamos a convertir nuestra instancia local en global ya que el modelo usuario se usará en varios métodos del archivo usuarioControl.js    Observa que la instancio usuario se cambió del método registrarUsuario hacia arriba  Desarrollar un nuevo método para el uso del login dentro del usuarioControl.js crearemos un método llamado accesoUsuarios, que pueda comprobar el email y la contraseña del sistema, el método recibe una petición en Json del usuario y genera una respuesta.  Con req.body parser, los parámetros que son enviados son convertidos a un objeto en Json.  function **accesoUsuario**(req, res) {      var params = req.body;      var email = params.email;      var password = params.password;      usuariosModelo.**findOne**({ email: email}, (err, user) => {          if (err) {              res.**status**(500).**send**({ mesagge: 'Error en la peticion' });          } else {          }      });  Con la función findOne podemos buscar un registro en la base de datos “usuarioModelo” por algún campo asignado, en este caso email, nos genera el método un callback en el que podemos tratar el error el hallazgo. Agrega el siguiente código en el que verifica en la primera estructura condicional “if” si el usuario no existe, para lo cual manda un error 404.       if (!user) {                  res.**status**(404).**send**({ mesagge: 'El usuario no existe' });              } else {                  bcrypt.**compare**(password, usuario.password, function(err, check) {                      if (check) {  *//devolver los datos del ususario logeado*                          console.**log**('coincide el password')                          if (params.gethash) {  *//devolver un token de jwt*                          } else {                              res.**status**(200).**send**({ user: user });                          }                      } else {                          res.**status**(404).**send**({ mesagge: 'El usuario no  se ha identificado' });                      }                  });              }  Si el usuario existe, ejecuta el método comparar de bycript entre el password ingresado con el almacenado, este método genera un callback con el error y la verificación “check”. En la estructura condicional “if” interior verifica si existe un “check” para devolver los datos del usuario logeado, y en la siguiente condición “if” devuelve un token de jwt que analizaremos en el siguiente capítulo, de lo contrario devuelve status 200, es decir, un usuario. Resumiendo creamos un acceso por password a través de un servicio encriptado que permite más adelante usar tokens con hash(datos del usuario codificados) de jwt con un middleware que genere la rutas para las peticiones.  Por ahora verificar que el método **accesoUsuario** funciona, para ello creamos una nueva ruta, en usuarioRuta.js como se observa:  'use strict'  var express = **require**('express');  var usuarioControl = **require**('../control/usuarioControl');  var api = express.**Router**();  api.**get**('/probando-control', usuarioControl.prueba);  api.**post**('/registrar', usuarioControl.registrarUsuario);  api.**post**('/login', usuarioControl.accesoUsuario);  module.exports = api;  Pero, recuerda que tienes que exportar **accesoUsuario,** no se te olvide hacerlo  module.exports = {  **prueba**,  **registrarUsuario**,  **accesoUsuario**  };  Guarda los dos archivos usuarioRuta.js y usuarioControl.js y ejecuta con npm start | |
| **Resultados**        **Creando el servicio para JWT** | |
| **Conclusiones:**  Realiza tus conclusiones una cuartilla completa | |
| **Referencias bibliográficas:**  Anota la referencias de tres libros que correspondan al tema | |
| **Anexos** | |

**FECHA DE ELABORACIÓN: 01-09-2021**