<table>
<tr><td colspan="3">NOMBRE Y APELLIDOS:
Kilian Jesús Sánchez Sánchez</td><td colspan="3">FECHA: 11-09-2023</td></tr>
<tr><td colspan="3">DOCENTE: MANUEL MACÍAS PÉREZ</td><td colspan="3">NOTA:</td></tr>
<tr><td colspan="3">(IFCD0210) DESARROLLO DE APLICACIONES CON TECNOLOGÍAS WEB.</td><td colspan="3">Nº CURSO: 22-35/008902</td></tr>
<tr><td>MF:</td><td>0492</td><td rowspan="3">UNIDADES DE APRENDIZAJE A LAS QUE RESPONDE:</td><td rowspan="3">UA1</td><td rowspan="3">Duración:</td><td rowspan="3">3 h</td></tr>
<tr><td>UF:</td><td>1845</td></tr>
<tr><td>PRÁCTICA Nº:</td><td>E1</td></tr>
<tr><td colspan="6">DENOMINACIÓN: Ficheros de datos.</td></tr>
</table>

DESCRIPCIÓN

1.- El alumno de forma individual deberá realizar un proyecto de una aplicación web incluyendo Ficheros de datos NoSQL.

- Crear una nueva base de datos NOSQL. (MongoDB).
- Crear la estructura de datos en el servidor. Según el siguiente modelo.
  - Código usuario
  - Nombre usuario
  - Código de la tarea a realizar
  - Descripción de la tarea a realizar
  - Puntuación
  - Estado
- Conectar el servidor Nodejs con la base de datos. (Seguir el esquema de clase)

Pegar en este Archivo el pantallazo de la base de datos creada en la aplicación Compass de MongoDB.
Adjuntar el código del modelo creado con la estructura de datos de la prueba y convertir este documento en pdf.
Enviar o Subir a Github.
La práctica se realizará de manera individual.

MEDIOS PARA SU REALIZACIÓN

- Equipo informático.
- Aplicación Visual Code Studio instalada en el equipo.
- Navegadores actualizados

PAUTAS DE ACTUACIÓN DEL FORMADOR

Al inicio de la práctica, que se desarrollará de manera individual por cada uno de los alumnos, el formador/a realizará las siguientes actuaciones:

- Fijará los objetivos de la práctica.

- Aportará las instrucciones necesarias a los alumnos/as para la realización de la misma, haciendo hincapié en aquellos aspectos más relevantes.

- Facilitará a cada alumno/a la documentación necesaria para el desarrollo de la práctica.

- Resolverá las dudas que se planteen durante el transcurso de la práctica, con objeto de que el alumnado aprenda y pueda concluir la realización de la misma.

Durante la realización de la práctica el formador/a supervisará el desarrollo de esta para evaluar tanto los procedimientos como el resultado final.

Al finalizar la práctica el formador examinará el desarrollo que han realizado los/as alumnos/as, proponiendo las medidas de corrección, en caso necesario.

ESPECIFICACIONES PARA LA EVALUACIÓN DE LA PRÁCTICA

Resultados a comprobar	Indicadores de logro
1. Crear componentes software utilizando objetos o componentes de conectividad específicos para acceder a informaciones almacenadas en bases de datos y otras estructuras. Conforme el criterio de evaluación CE 1.1	1.1 Crea componentes software utilizando objetos de conectividad específicos para acceder a informaciones almacenadas en bases de datos y otras estructuras
	1.2 Crea componentes software utilizando componentes de conectividad específicos para acceder a informaciones almacenadas en bases de datos y otras estructuras.
	1.3 Aplica los componentes software creados para acceder a informaciones almacenadas en bases de datos y otras estructuras.
	1.4 Documenta los componentes software creados para acceder a informaciones almacenadas en bases de datos y otras estructuras.
	1.5 Entiende los componentes software creados para acceder a informaciones almacenadas en bases de datos y otras estructuras.

Sistema de valoración

Definición de indicadores y escalas de medida

Los indicadores que se van a establecer, será una hoja de chequeo, sistema de valoración, que complementa a este documento, donde se evalúan todos los resultados a comprobar (tareas). En este documento, se establecerán a su vez los indicadores de logro que se han de tener en cuenta, para conseguir los resultados a comprobar.

Mínimo exigible

El mínimo exigible para la superación de la práctica es de 50 puntos sobre 100 puntos

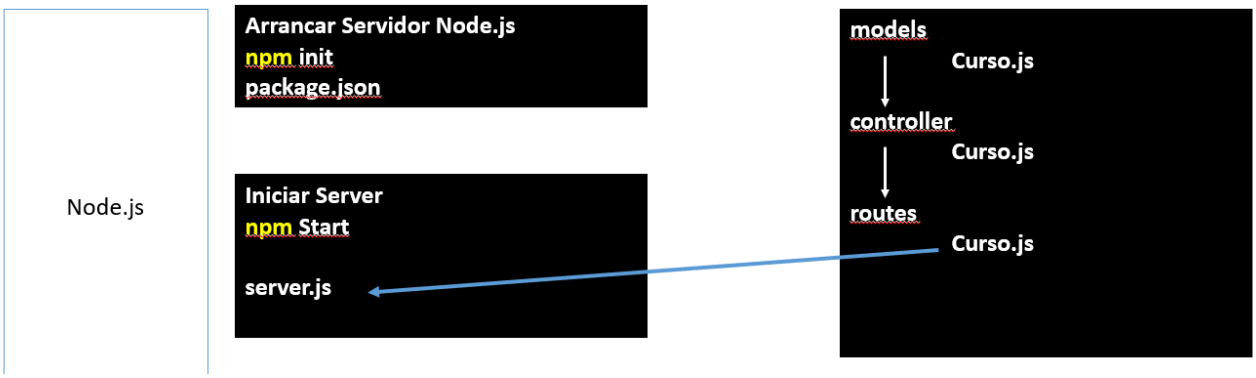
SUPUESTO PRÁCTICO

1.- El alumno de forma individual deberá realizar un proyecto de una aplicación web incluyendo Ficheros de datos NoSQL.

- Crear una nueva base de datos NOSQL. (MongoDB).
- Crear la estructura de datos en el servidor. Según el siguiente modelo.
    - Código usuario
    - Nombre usuario
    - Código de la tarea a realizar
    - Descripción de la tarea a realizar
    - Puntuación
    - Estado
- Conectar el servidor Nodejs con la base de datos. (Seguir el esquema de clase)

Pegar en este Archivo el pantallazo de la base de datos creada en la aplicación Compass de MongoDB, Adjuntar el código del modelo creado con la estructura de datos de la prueba y convertir este documento en pdf.
Enviar o Subir a Github.
La práctica se realizará de manera individual.

Ejemplo:




```
const mongoose =  require('mongoose')

const UserSchema = new mongoose.Schema(
    {
    cod: Number,
    nombre: String,
    apellidos: String,
    edad: Number,
    mail: String
    }
)

module.exports = mongoose.model('user', UserSchema)
```

```
const Model = require('../models/mCurso')
const fs = require('fs');
const path =  require('path');
const controller ={
    saveCurs: function(req, res){
        const curs = new Model();
        const params = req.body;
        curs.nombreU = params.nombreU;
        curs.descripC= params.descripC;
        curs.tarea = params.tarea;
        curs.estado  = params.estado;
        curs.puntaje= params.puntaje;
        curs.estado = params.estado;
        Curssave((err, cursStored)=>{
            if(err) return res.status(500).send({message: 'Error al guardar el documento'})
            if(!projectStored) return res.status(404).send({menssage:'No se a podido'})
            return res.status(200).send({curso:cursStored});
        });
        },
        getCurs: function(req, res){
            var cursId = req.params.codigo;
            if(cursId==null) return res.status(404).send({menssage:'El curso no existe'})

            User.findById(cursID, (err, curs)=>{
                if(err) return res.status(500).send({menssage: 'error al devolver los datos'});
                if(!curs) return res.status(404).send({mensaje: 'el curso no existe'});
                return res.status(200).send({
                    curs
                })
            })
        },
        getCursos: function(req, res){
            User.find({}).sort('-year').exec((err, curs)=>{
                if(err) return res.status(500).send({message: 'error al devolver datos'});
                if(!projects) return res.status(404).send({menssage: 'no hay curso que mostrar'});
                return res.status(200).send({project});
            })
        },
        updateCurs: function(req, res){
            var cursId = req.params.codigo;
            var update = res.body;
            User.findBuIdAndUpdate(cursId, update, {new:true}, (err, cursUpdated)=>{
                if(err) return res.status(500).send({message: 'Error al actualizar'});
                if(!cursUpdated) return res.status(404).send({menssage: 'no existe el curso para actualizar'});
                return res.status(200).send({
                    curs:cursUpdated
                })
            })
        },
        deleteCurs: function(req, res){
            var cursId = req.params.codigo;
            User.findByIdAndRemove(cursId,(err, cursRemoved)=>{
                if(err) return res.status(500).send({message: 'no se ha podido borrar el curso'});
                if(!cursRemoved) return res.status(404).send({message: ' no se puede eliminar ese curso'});
                return res.status(200).send({
                    curs: cursRemoved
                });
            });
        },
    };
    module.exports = controller;
```

```
const mongoose = require('mongoose')

const CursSchema = new mongoose.Schema(
    {
        codigo: {
            type: String
        },
        nombreU:{
            type: String
        },
        tarea:{
            type: Number
        },
        descripC:{
            type: String
        },
        puntaje: {
            type: Number
        },
        estado:{
            type: Boolean
        }
    }
)
module.exports = mongoose.model('curso', CursSchema)
```

```
const express = require('express')
const CursController = require('../controllers/cCurso')
const router = express.Router()

router.post('/save-curs', CursController.saveCurs);
router.get('/curs/:id?', CursController.getCurs);
router.get('/curso', CursController.getCursos);
router.put('/curs/:id', CursController.updateCurs);
router.delete('/curs/:id', CursController.deleteCurs);


module.exports = router
```

```
1  const express = require('express')
2
3
4  const app = express()
5
6  const port = 3700
7
8  app.listen(port, ()=>{
9      console.log('La aplicacion esta en linea')
10 })
11 const userRouters = require('./routes/user')
12 const CursosRouters = require('./routes/cursos')
13 const CursRouters = require('./routes/curso')
14
15
16
17 app.use(userRouters)
18
19 app.use(CursosRouters)
20
21 app.use(CursRouters)
22
23
24
25 const mongoose = require('mongoose')
26
27 mongoose.Promise = global.Promise
28
29 mongoose.connect('mongodb://127.0.0.1:27017/Appweb', {
30     useNewUrlParser: true,
31 })
32 .then(()=>{
33     console.log("Conexion establecida...")
34 })
35 .catch(err => console.log(err))
```


My Queries
Databases
Search
Appweb
Curso
Cursos
Usuarios
cursos
users
admin
config
local
Appweb.Curso
1
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' }
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1-1 of 1
_id: Object
nombreU: ""
tarea: ""
descripC: ""
puntaje: ""
estado: ""

SISTEMAS DE VALORACIÓN MF 0492_3 – UF1845 – E1

RESULTADOS A COMPROBAR	INDICADORES DE LOGRO	ESCALA DE MEDIDAS		
1. Crear componentes software utilizando objetos o componentes de conectividad específicos para acceder a informaciones almacenadas en bases de datos y otras estructuras. Conforme el criterio de evaluación CE 1.1	1.1 Crea componentes software utilizando objetos de conectividad específicos para acceder a informaciones almacenadas en bases de datos y otras estructuras	- Crea componentes software utilizando objetos de conectividad específicos entre un 75% y 100% - Crea componentes software utilizando objetos de conectividad específicos entre un 50 % y 75% - Crea componentes software utilizando objetos de conectividad específicos por debajo de un 50 %	B R M	20 10 0
	1.2 Crea componentes software utilizando componentes de conectividad específicos para acceder a informaciones almacenadas en bases de datos y otras estructuras.	- Crea componentes software utilizando componentes de conectividad específicos entre un 75% y 100%. - Crea componentes software utilizando componentes de conectividad específicos entre un 50% y 75%. - Crea componentes software utilizando componentes de conectividad específicos por debajo de un 50%.	B R M	20 10 0
	1.3 Aplica los componentes software creados para acceder a informaciones almacenadas en bases de datos y otras estructuras.	- Aplica los componentes software creados para acceder a informaciones almacenadas entre un 75% y 100%. - Aplica los componentes software creados para acceder a informaciones almacenadas entre un 50% y 75%. - Aplica los componentes software creados para acceder a informaciones almacenadas por debajo de un 50%.	B R M	20 10 0
	1.4 Documenta los componentes software creados para acceder a informaciones almacenadas en bases de datos y otras estructuras.	- Documenta los componentes software creados para acceder a informaciones entre un 75% y 100%. - Documenta los componentes software creados para acceder a informaciones entre un 50% y 75%. - Documenta los componentes software creados para acceder a informaciones por debajo de un 50%.	B R M	20 10 0
	1.5 Entiende los componentes software creados para acceder a informaciones almacenadas en bases de datos y otras estructuras.	- Entiende los componentes software creados para acceder a informaciones entre un 75% y 100%. - Entiende los componentes software creados para acceder a informaciones entre un 50% y 75%. - Entiende los componentes software creados para acceder a informaciones por debajo de un 50%	B R M	20 10 0
	Valor mínimo exigible: 50	**Valor máximo: 100**		