NOMBRE Y APELLIDOS:			**FECHA: 18-09-2023**		
DOCENTE: MANUEL MACÍAS PÉREZ			**NOTA:**		
(IFCD0210) DESARROLLO DE APLICACIONES CON TECNOLOGÍAS WEB.			**Nº CURSO: 22-35/008902**		
MF:	0492	UNIDADES DE APRENDIZAJE A LAS QUE RESPONDE:	UA	Duración:	3 h
UF:	1845				
PRÁCTICA Nº:	E2				

DENOMINACIÓN: **Ficheros de datos.**

DESCRIPCIÓN

1.- El alumno de forma individual deberá realizar lo siguiente:

- Crear un componente en Angular para enviar y recibir datos desde la base de datos creada en la prueba E1.
- Crear su vista plantilla en html
- Dar estilos con CSS o utilizar Angular Material.

Pegar en este Archivo el pantallazo de cómo se vería la vista en un navegador.
Pegar en este Archivo el pantallazo de la base creada en E1 con al menos un objeto guardado.
Adjuntar el código del archivo TypeScript del componente creado 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. Integrar sentencias en los componentes software para acceder y manipular la información ubicada en bases de datos Conforme el criterio de evaluación CE 1.2	1.1 Integra sentencias en los componentes software para acceder a la información ubicada en bases de datos
	1.2 Manipula la información ubicada en bases de datos
2. En un supuesto práctico en el que se pide construir componentes de software que accedan a datos soportados en bases de datos u otras estructuras de almacenamiento. Conforme el criterio de evaluación CE 1.3	2.1 Identifica los elementos y estructuras contenidas en una base de datos.
	2.2 Utiliza los objetos, conectores y middleware necesarios en la construcción del componente para realizar los accesos a los datos soportados en la base de datos
	2.3 Realiza operaciones de definición y manipulación de informaciones soportadas en bases de datos

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

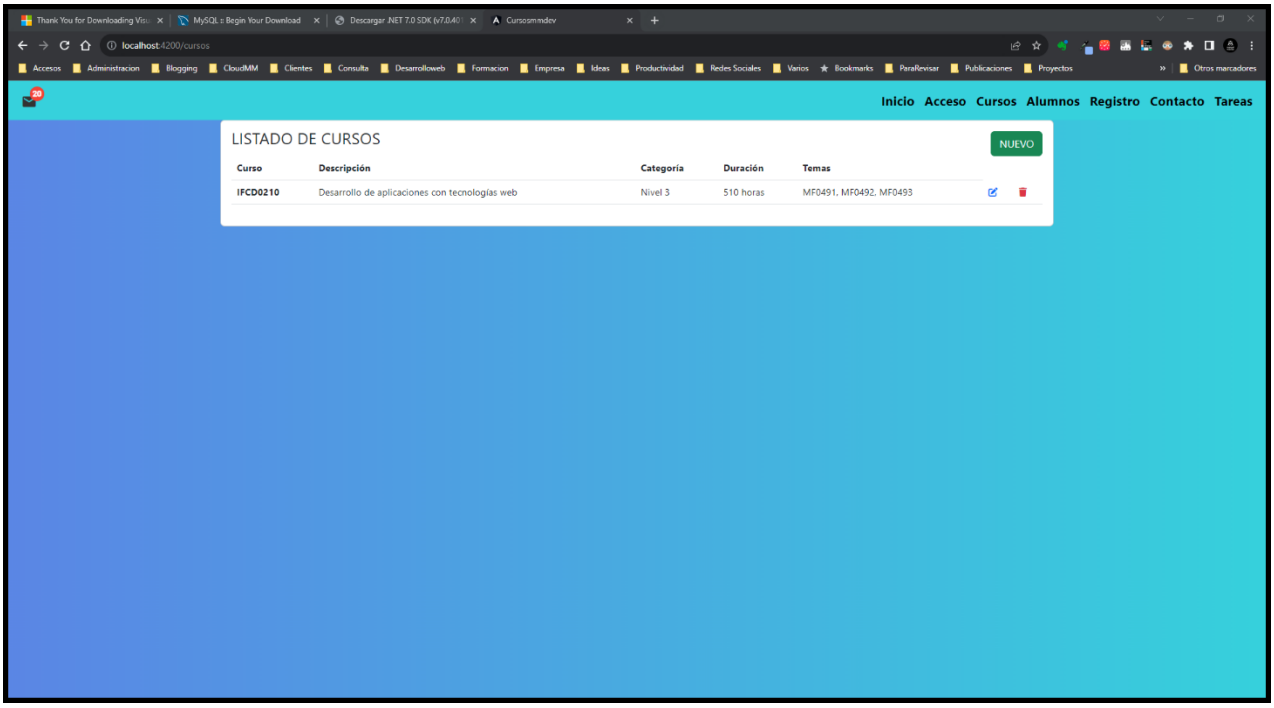
1.- El alumno de forma individual deberá realizar lo siguiente:

- Crear un componente en Angular para enviar y recibir datos desde la base de datos creada en la prueba E1.
- Crear su vista plantilla en html
- Dar estilos con CSS o utilizar Angular Material.

Pegar en este Archivo el pantallazo de cómo se vería la vista en un navegador.
Pegar en este Archivo el pantallazo de la base creada en E1 con al menos un objeto guardado.
Adjuntar el código del archivo TypeScript del componente creado y convertir este documento en pdf. Enviar o Subir a Github.
La práctica se realizará de manera individual.

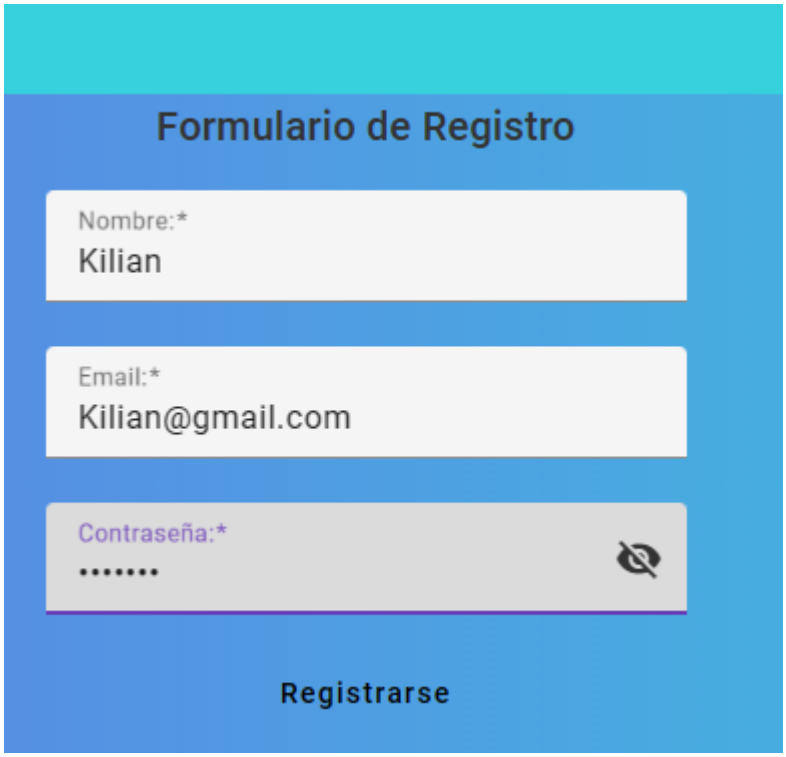
Ejemplo:

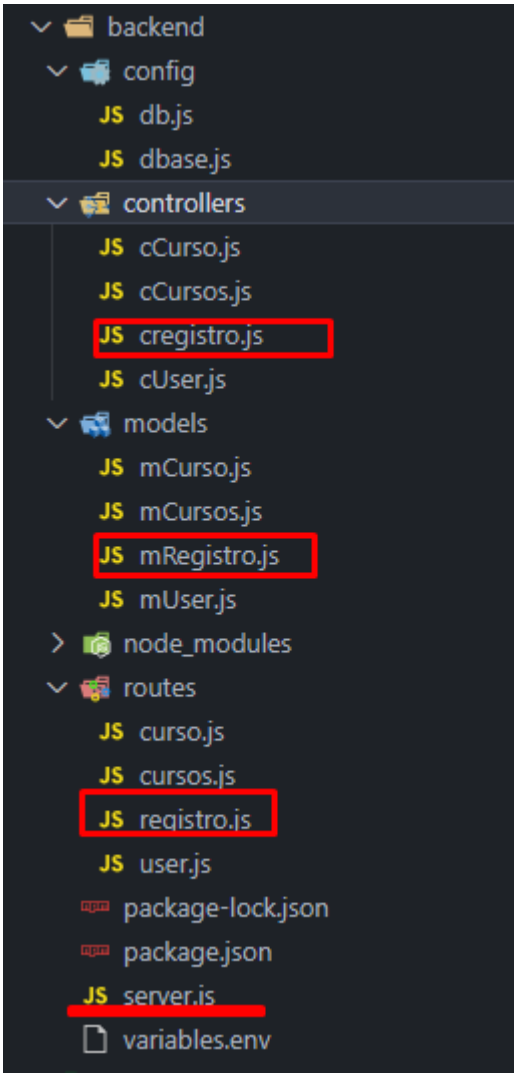
```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { GCurso } from '../Models/gcurso';
@Injectable({
  providedIn: 'root'
})
export class GcursoService {
  url= 'http://localhost:3700/api/cursos/'
  constructor(private http: HttpClient) { }
  getCursos(): Observable<any>{
    return this.http.get(this.url)}
  eliminarCurso(id: string): Observable<any> {
    return this.http.delete(this.url + id);}
  guardarCurso(curso: GCurso): Observable<any> {
    return this.http.post(this.url, curso);}
  obtenerCurso(id: string): Observable<any> {
    return this.http.get(this.url + id);}
  editarCurso(id: string, curso: GCurso): Observable<any>{
    return this.http.put(this.url + id, curso)}
}
```

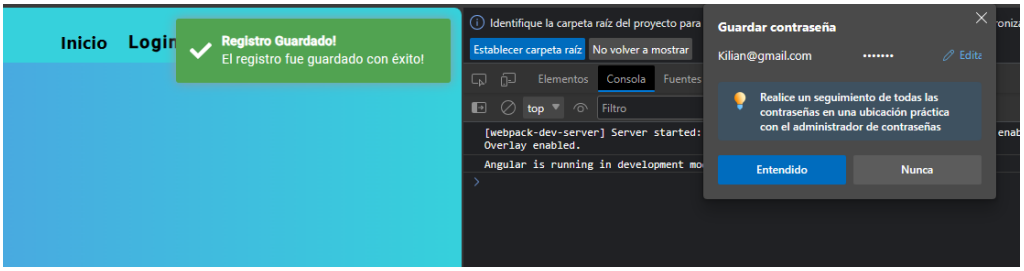


```html
<div class="Form">
    <h2>Formulario de Registro</h2>
    <form [formGroup]="registroForm" (ngSubmit)="agregarRegistro()">

        <mat-form-field appearance="fill">
        <mat-label for="nombre">Nombre:</mat-label>
        <input  matInput type="text" id="nombre" formControlName="nick" name="nombre" required></mat-form-field>

          <mat-form-field appearance="fill">
        <mat-label for="email">Email:</mat-label>
        <input matInput placeholder="pat@ejemplo.com" type="email" id="email" formControlName="email" name="email" required>
      </mat-form-field>
        <mat-form-field appearance="fill">
        <mat-label for="contrasena">Contraseña:</mat-label>
        <input matInput [type]="hide ? 'password' : 'text'" id="contrasena" formControlName="pass" name="contrasena" required>
        <button mat-icon-button matSuffix (click)="hide = !hide" [attr.aria-label]="'Hide password'" [attr.aria-pressed]="hide">
          <mat-icon>{{hide ? 'visibility_off' : 'visibility'}}</mat-icon>
        </button>
      </mat-form-field>

      <button mat-button [disabled]="registroForm.invalid" type="submit">Registrarse</button>

    </form>
  </div>

```

```typescript
import { Component, OnInit } from '@angular/core'
import {FormBuilder, FormGroup, Validators} from '@angular/forms'
import {ActivatedRoute, Router} from '@angular/router'
import { ToastrService } from 'ngx-toastr'

import { Registrad } from '../Models/registro'
import { GRegistroService } from '../servicios/gregistro.service'

@Component({
  selector: 'app-registro',
  templateUrl: './registro.component.html',
  styleUrls: ['./registro.component.css']
})
export class RegistroComponent implements OnInit{
  registroForm: FormGroup;
  titulo = 'Crear curso';
    id: string | null;
    constructor(private fb: FormBuilder,
                private router: Router,
                private toastr: ToastrService,
                private _RegistroService: GRegistroService,
                private aRouter: ActivatedRoute) {
      this.registroForm = this.fb.group({
        nick: ['', Validators.required],
        email: ['', Validators.required],
        pass: ['', Validators.required]
      })
      this.id = this.aRouter.snapshot.paramMap.get('id');
    }

  ngOnInit(): void {

  }
  agregarRegistro() {
    const REGISTRO: Registrad = {
      nick: this.registroForm.get('nick')?.value,
      email: this.registroForm.get('email')?.value,
      pass: this.registroForm.get('pass')?.value
    }

    if (REGISTRO.nick !== '' && REGISTRO.email !== '' && REGISTRO.pass !== '') {
      // Realizar acciones para guardar el registro en la base de datos
      this._RegistroService.guardarRegistro(REGISTRO).subscribe(data => {
        this.toastr.success('El registro fue guardado con éxito!', 'Registro Guardado!');
        this.router.navigate(['/']); // Redirigir a otra página después de guardar el registro
      }, error => {
        console.log(error);
        this.registroForm.reset();
      });
    } else {
      // Realizar acciones si faltan campos requeridos en el formulario
      this.toastr.error('Por favor, complete todos los campos requeridos.', 'Error');
    }
  }
hide = true;
}
```


Formulario de Registro
Nombre:*
Kilian
Email:*
Kilian@gmail.com
Contraseña:*
•••••••
Registrarse


backend
config
db.js
dbase.js
controllers
cCurso.js
cCursos.js
cregistro.js
cUser.js
models
mCurso.js
mCursos.js
mRegistro.js
mUser.js
node_modules
routes
curso.js
cursos.js
registro.js
user.js
package-lock.json
package.json
server.js
variables.env


Inicio
Registro Guardado!
El registro fue guardado con éxito!
Identifique la carpeta raíz del proyecto para
Establecer carpeta raíz
No volver a mostrar
Elementos
Consola
Fuentes
top
Filtro
[webpack-dev-server] Server started:
Overlay enabled.
Angular is running in development mo
Guardar contraseña
Kilian@gmail.com
•••••••
Edite
Realice un seguimiento de todas las contraseñas en una ubicación práctica con el administrador de contraseñas
Entendido
Nunca

```
const Registro = require("../models/mRegistro");

exports.crearRegistro = async (req, res) =>{
    try{
        let registro;
        registro = new Registro(req.body);
        await registro.save();
        res.send(registro);
    }catch (error){
        console.log(error);
        res.status(500).send('Hubo un error')
    }
}

exports.obtenerRegistro = async(req, res) => {
    try{
        let registro = await Registro.findById(req.params.id);
        if(!registro){
            res.status(404).json({msg:'No existe el Registro'})
        }
        res.json(registro);
    }catch(error){
        console.log(error);
        res.status(500).send('Hubo un error');
    }
}

exports.obtenerRegistros = async (req, res) => {
    try {
        const registros = await Registro.find();
        res.json(registros)
    } catch (error) {
        console.log(error);
        res.status(500).send('Hubo un error');
    }
}
```

```
const RegistroSchema = mongoose.Schema({
    nick:{
        type: String,
        required: true
    },
    email:{
        type: String,
        required: true
    },
    pass:{
        type: String,
        required: true
    }
});

module.exports = mongoose.model('Registro', RegistroSchema);
```

```
// Rutas para registro
const express = require('express');
const router = express.Router();
const registroController = require('../controllers/cregistro');

// api/registro
router.post('/', registroController.crearRegistro);
router.get('/', registroController.obtenerRegistros);
// router.put('/:id', registroController.actualizarRegistro);
router.get('/:id', registroController.obtenerRegistro);
// router.delete('/:id', registroController.eliminarRegistro);

module.exports = router;
```

SISTEMAS DE VALORACIÓN MF 0492_3 – UF1845 – E2

RESULTADOS A COMPROBAR	INDICADORES DE LOGRO	ESCALA DE MEDIDAS		
3. Integrar sentencias en los componentes software para acceder y manipular la información ubicada en bases de datos Conforme el criterio de evaluación CE 1.2	Integra sentencias en los componentes software para acceder a la información ubicada en bases de datos	- Integra sentencias en los componentes software para acceder a la información ubicada en bases de datos entre un 75% y 100% - Integra sentencias en los componentes software para acceder a la información ubicada en bases de datos entre un 50 % y 75% - Integra sentencias en los componentes software para acceder a la información ubicada en bases de datos por debajo de un 50 %	B R M	20 10 0
	Manipula la información ubicada en bases de datos	- Manipula la información ubicada en bases de datos entre un 75% y 100%. - Manipula la información ubicada en bases de datos entre un 50% y 75%. - Manipula la información ubicada en bases de datos por debajo de un 50%.	B R M	20 10 0
4. En un supuesto práctico en el que se pide construir componentes de software que accedan a datos soportados en bases de datos u otras estructuras de almacenamiento. Conforme el criterio de evaluación CE 1.3	2.1 Identifica los elementos y estructuras contenidas en una base de datos.	- Identifica los elementos y estructuras contenidas en una base de datos entre un 75% y 100%. - Identifica los elementos y estructuras contenidas en una base de datos entre un 50% y 75%. - Identifica los elementos y estructuras contenidas en una base de datos por debajo de un 50%.	B R M	20 10 0
	2.2 Utiliza los objetos, conectores y middleware necesarios en la construcción del componente para realizar los accesos a los datos soportados en la base de datos	- Utiliza los objetos, conectores y middleware necesarios en la construcción del componente entre un 75% y 100%. - Utiliza los objetos, conectores y middleware necesarios en la construcción del componente entre un 50% y 75%. - Utiliza los objetos, conectores y middleware necesarios en la construcción del componente por debajo de un 50%.	B R M	20 10 0
	2.3 Realiza operaciones de definición y manipulación de informaciones soportadas en bases de datos	- Realiza operaciones de definición y manipulación de informaciones soportadas en bases de datos entre un 75% y 100%. - Realiza operaciones de definición y manipulación de informaciones soportadas en bases de datos entre un 50% y 75%. - Realiza operaciones de definición y manipulación de informaciones soportadas en bases de datos por debajo de un 50%.	B R M	20 10 0
	Valor mínimo exigible: 50	**Valor máximo: 100**		