

NOMBRE Y APELLIDOS: Daniel García Molina			FECHA: 25-08-2023		
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:	UA2	Duración:	3 h
UF:	1844				
PRÁCTICA Nº:	E2				
DENOMINACIÓN: Principios de la orientación a objetos					
<p>DESCRIPCIÓN</p> <p>1.- El alumno de forma individual deberá en base a la planificación realizada en la prueba E1 de la UF1844, realizar lo siguiente:</p> <ul style="list-style-type: none"> Adaptar la documentación basándose en las herramientas vistas en clase (Framework), tomando como referencia la programación por componentes del FrameWork de desarrollo. Crear el proyecto de aplicación con sus componentes definido en la documentación. Incluir el documento creado para la planificación de la aplicación en formato .md dentro del directorio raíz del proyecto. <p>Pegar en este Archivo el resultado de la prueba y convertir en pdf. Enviar o Subir a Github. La práctica se realizará de manera individual.</p> <p>MEDIOS PARA SU REALIZACIÓN</p> <ul style="list-style-type: none"> Equipo informático. Aplicación Visual Code Studio instalada en el equipo. Navegadores actualizados <p>PAUTAS DE ACTUACIÓN DEL FORMADOR</p> <p><i>Al inicio de la práctica, que se desarrollará de manera individual por cada uno de los alumnos, el formador/a realizará las siguientes actuaciones:</i></p> <ul style="list-style-type: none"> 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. <p>Durante la realización de la práctica el formador/a supervisará el desarrollo de esta para evaluar tanto los procedimientos como el resultado final.</p> <p>Al finalizar la práctica el formador examinará el desarrollo que han realizado los/as alumnos/as, proponiendo las medidas de corrección, en caso necesario.</p>					
ESPECIFICACIONES PARA LA EVALUACIÓN DE LA PRÁCTICA					

Resultados a comprobar	Indicadores de logro
<p>1. Crea objetos, clases y métodos adecuados a la funcionalidad del componente software a desarrollar utilizando lenguajes de programación orientados a objetos.</p> <p>Conforme el criterio de evaluación CE 1.2</p>	1.1 Crea objetos adecuados a la funcionalidad del componente software a desarrollar utilizando lenguajes de programación orientados a objetos
	1.2 Crea clases adecuadas a la funcionalidad del componente software a desarrollar utilizando lenguajes de programación orientados a objetos
	1.3 Crea métodos adecuados a la funcionalidad del componente software a desarrollar utilizando lenguajes de programación orientados a objetos

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 en base a la planificación realizada en la prueba E1 de la UF1844, realizar lo siguiente:

- Adaptar la documentación basándose en las herramientas vistas en clase (Framework), tomando como referencia la programación por componentes del Framework de desarrollo.
- Crear el proyecto de aplicación con sus componentes definido en la documentación.
- Incluir el documento creado para la planificación de la aplicación en formato .md dentro del directorio raíz del proyecto.

Pegar en este Archivo el resultado de la prueba y convertir en pdf. Enviar o Subir a Github.

La práctica se realizará de manera individual.

Ejemplo:

- Interacción del usuario:
 - A través de botones en la parte superior – derecha:
 - Botón 1 – Acceder
 - .
 - .
 - .
 - A través de formularios de pedidos donde se recogerán los siguientes datos
 - Formulario de pedido
 - Nombre
 - Correo electrónico
 - Teléfono
 - Formulario de consulta
 - .
 - .
 - .

Aplicación para la asignación de Tareas en el entorno educativo “TaskLord”.

1. Interacción del usuario:

o A través de botones en la parte superior – derecha:

Inicio.

Alumnos. (**AlumnosComponent**) Nos aparece un listado de los distintos alumnos, cada uno con un botón para ver información concreta sobre cada uno. **AlumnoComponent**

Profesores. **ProfesoresComponent** Nos aparece un listado de los distintos profesores, cada uno con un botón para ver información concreta sobre cada uno. **ProfesorComponent**

Cursos. (**CursosComponent**) Nos aparecen los distintos cursos disponibles para la matriculación inscripción, cada uno con un botón para ver información sobre el curso en concreto. **CursoComponent**

Acceso. (**AccesoComponent**) Nos aparece un formulario de Login.

Registro. (**RegistroComponent**) Nos aparece un formulario de Registro.

Contacto. (**ContactoComponent**) Nos aparece un formulario de contacto.

o A través de formularios donde se recogerán los siguientes datos :

- Formulario de Acceso:
 - Usuario.
 - Correo electrónico.
 - Contraseña.

- Formulario de Registro:
 - Nombre.
 - Apellidos.
 - Correo electrónico.
 - Edad.
 - Contraseña.
 - Url Imagen.

- Formulario de Contacto:

- Nombre.
- Correo electrónico.
- Mensaje.

2. Cómo procesamos los datos y cómo los transformamos:

a) **Almacenar los datos:** Los Datos introducidos por el usuario en los formularios de Contacto, Registro y Acceso serán guardados con el comando (ngSubmit) en la etiqueta <form> de cada formulario mediante el componente Form de Angular.

b) **Validación de datos:** Los Datos introducidos por el usuario tendrán una doble validación, por parte de la aplicación y por parte del servidor.

c) **Proceso de datos:** Cada formulario tendrá en el servidor una base de datos asociada a ese formulario dónde se guardarán y almacenarán los datos introducidos por los usuarios y validados doblemente.

3. Salida de Datos.

- Notificación de alumno nuevo en la esquina superior derecha en el “botón” “Alumnos”
- En la zona principal de la aplicación saldrán cuadros con los datos, listados o formularios correspondientes.

Código HTML del archivo app.component:

```
<!DOCTYPE html>

<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>TaskLord</title>
</head>
<body>
```

```

<nav>
  <h3>TaskLord</h3>
  <ul>
    <a href=""></a>
    <li><a routerLink="">Inicio</a></li>
    <li><a routerLink="/alumnos">Alumnos</a></li>
    <li><a routerLink="/profesores">Profesores</a></li>
    <li><a routerLink="/cursos">Cursos</a></li>
    <li><a routerLink="/acceso">Acceso</a></li>
    <li><a routerLink="/registro">Registro</a></li>
    <li><a routerLink="/contacto">Contacto</a></li>
  </ul>
</nav>

<main>
  <router-outlet></router-outlet>
  <ng-template>
  </ng-template>
</main>
</body>
</html>

```

Código CSS del archivo app.component:

```

@import url('https://fonts.googleapis.com/css2?
family=Quicksand:wght@300;400;500;600;700&display=swap');

```

```

*{
  box-sizing: border-box;
  margin: 0;
  padding: 0;
  font-family: 'Quicksand', sans-serif;
}

```

```

main{
  padding: 3rem;
}

```

```

h3{
  background-color: #00214A;
}

```

```

color: #cdeae8;
text-align: center;
width: 25vw;
padding: 0.4rem;
}

nav{
display: flex;
flex-direction: row;
justify-content: space-between;
background-color: #cdeae8;
margin: 0;
padding: 0;
}

ul{
display: flex;
flex-direction: row;
list-style: none;
align-items: center;
}

li{
margin-left: 2rem;
padding-right: 2rem;
}

a{
text-decoration: none;
color: #00214A;
}

```

Código TS del archivo app.component:

```

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { AccesoComponent } from './acceso/acceso.component';
import { RegistroComponent } from './registro/registro.component';

```



```
import { ContactoComponent } from './contacto/contacto.component';
import { AlumnoComponent } from './alumno/alumno.component';
import { AlumnosComponent } from './alumnos/alumnos.component';
import { CursoComponent } from './curso/curso.component';
import { CursosComponent } from './cursos/cursos.component';
import { ProfesorComponent } from './profesor/profesor.component';
import { ProfesoresComponent } from './profesores/profesores.component';
```

```
@NgModule({
  declarations: [
    AppComponent,
    AccesoComponent,
    RegistroComponent,
    ContactoComponent,
    AlumnoComponent,
    AlumnosComponent,
    CursoComponent,
    CursosComponent,
    ProfesorComponent,
    ProfesoresComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

SISTEMAS DE VALORACIÓN MF 0492_3 – UF1844 – E2

RESULTADOS A COMPROBAR	INDICADORES DE LOGRO	ESCALA DE MEDIDAS		
<p>2. Crea objetos, clases y métodos adecuados a la funcionalidad del componente software a desarrollar utilizando lenguajes de programación orientados a objetos.</p> <p>Conforme el criterio de evaluación CE 1.2</p>	Crea objetos adecuados a la funcionalidad del componente software a desarrollar utilizando lenguajes de programación orientados a objetos	<p>- Crea objetos adecuados a la funcionalidad del componente software entre un 75% y 100%</p> <p>- Crea objetos adecuados a la funcionalidad del componente software entre un 50 % y 75%</p> <p>- Crea objetos adecuados a la funcionalidad del componente software por debajo de un 50 %</p>	B	40
			R	20
			M	0
	Crea clases adecuadas a la funcionalidad del componente software a desarrollar utilizando lenguajes de programación orientados a objetos	<p>- Crea clases adecuadas a la funcionalidad del componente software entre un 75% y 100%.</p> <p>- Crea clases adecuadas a la funcionalidad del componente software entre un 50% y 75%.</p> <p>- Crea clases adecuadas a la funcionalidad del componente software por debajo de un 50%.</p>	B	30
			R	15
			M	0
	Crea métodos adecuados a la funcionalidad del componente software a desarrollar utilizando lenguajes de programación orientados a objetos	<p>- Crea métodos adecuados a la funcionalidad del componente software entre un 75% y 100%.</p> <p>- Crea métodos adecuados a la funcionalidad del componente software entre un 50% y 75%.</p> <p>- Crea métodos adecuados a la funcionalidad del componente software por debajo de un 50%.</p>	B	30
			R	15
			M	0
	Valor mínimo exigible: 50	Valor máximo: 100		