Definición Entrega ET2
Interfaces de Usuario
Curso 2023-2024
—-DEFINITIVA—-

Competencias a evaluar

Código	Descripción	ET2
A4	Coñecementos básicos sobre o uso e programación dos ordenadores, sistemas operativos, bases de datos e programas informáticos con aplicación na enxeñería	
A23	Capacidade para deseñar e avaliar interfaces persoa-computador que garantan a accesibilidade e usabilidade aos sistemas, servizos e aplicacións informáticas	X
A25	Capacidade para desenvolver, manter e avaliar servizos e sistemas software que satisfagan todos os requisitos do usuario e se comporten de forma fiable e eficiente, sexan asequibles de desenvolver e manter e cumpran normas de calidade, aplicando as teorías, principios, métodos e prácticas da Enxeñería do Software	X
A26	Capacidade para valorar as necesidades do cliente e especificar os requisitos software para satisfacer estas necesidades, reconciliando obxectivos en conflito mediante a procura de compromisos aceptables dentro das limitacións derivadas do custo, do tempo, da existencia de sistemas xa desenvolvidos e das propias organizacións	X
A28	Capacidade de identificar e analizar problemas e deseñar, desenvolver, implementar, verificar e documentar solucións software sobre a base dun coñecemento axeitado das teorías, modelos e técnicas actuais	X
A33	Capacidade para empregar metodoloxías centradas no usuario e a organización para o desenvolvemento, avaliación e xestión de aplicacións e sistemas baseados en tecnoloxías da información que aseguren a accesibilidade, ergonomía e usabilidade dos sistemas	
B1	Capacidade de análise, síntese e avaliación	X
B2	Capacidade de organización e planificación	X
B3	Comunicación oral e escrita na lingua nativa	
B5	Capacidade de abstracción: capacidade de crear e utilizar modelos que reflictan situacións reais	X
B8	Resolución de problemas	X
B9	Capacidade de tomar decisións	X
B10	Capacidade para argumentar e xustificar loxicamente as decisións tomadas e as opinións	

B11	Capacidade de actuar autonomamente	X
B12	Capacidade de traballar en situacións de falta de información e/ou baixo presión	X
B13	Capacidade de integrarse rapidamente e traballar eficientemente en equipos unidisciplinares e de colaborar nun entorno multidisciplinar	
B15	Capacidade de relación interpersoal	
B16	Razoamento crítico	X
B18	Aprendizaxe autónoma	X
B19	Adaptación a novas situacións	
B20	Creatividade	X
B21	Liderazgo	
B22	Ter iniciativa e ser resolutivo	X

Tipología

Entrega individual de realización individual.

Definición

Dadas las tablas que se proporcionan en este punto y la definición de los formatos correctos de entrada de datos para cada uno de los atributos, se solicita la creación de la presentación de muestra de tuplas de cada tabla junto con los formularios de ADD, SEARCH, EDIT, DELETE y SHOWCURRENT con las pruebas de verificación de los formatos de cada campo y la definición de los test posibles a realizar para cada campo y la batería de pruebas que verifica la prueba de los test definidos anteriormente

```
--
-- Estructura de tabla para la tabla `programa`
--

CREATE TABLE `programa` (
  `id_programa` int(6) NOT NULL, AUTOINCREMENTAL
  `nombre_programa` varchar(60) NOT NULL,
  `acronimo_programa` varchar(20) NOT NULL,
  `nombre_original_programa` varchar(60) NOT NULL,
  `autor_programa` varchar(50) NOT NULL,
  `autor_original_programa` varchar(50) NOT NULL,
  `ano_programa` int(4) NOT NULL,
  `ano_original_programa` int(4) NOT NULL,
  `requisitos_programa` varchar(300) NOT NULL,
  `poblacion_desde_programa` int(2) NOT NULL,
  `poblacion_hasta_programa` int(2) NOT NULL,
  `unidad_poblacion` enum('MESES','AÑOS') NOT NULL,
    `tipo_programa` enum('EVALUACIÓN' , 'INTERVENCIÓN' , 'EVALUACIÓN E
```

```
INTERVENCIÓN') NOT NULL,
  `tiempo_aplicacion_programa` int(4) NOT NULL COMMENT 'tiempo de aplicación de programa en min',
   `descrip_interp_programa` varchar(5000) NOT NULL COMMENT 'descripcion e interpretación del programa',
  `fichero_programa` varchar(60) NOT NULL,
  `enlace_programa` varchar(100) NOT NULL,
  `formato_programa` enum('PAPEL','ELECTRÓNICO','PAPEL Y ELECTRÓNICO') NOT NULL,
`modo_correccion_programa` enum('PAPEL','ELECTRÓNICO','PAPEL Y ELECTRÓNICO') NOT NULL,
`modo_aplicacion_programa` enum('INDIVIDUAL','COLECTIVO','INDIVIDUAL Y COLECTIVO') NOT NULL,
  `imagen_programa` varchar(60) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Formatos permitidos

id_programa	dígitos min 1 max 6
nombre_programa	alfabéticos con acentos, ñ y espacios min 6 max 60
acronimo_programa	alfabéticos sin acentos ni espacios min 3 max 20
nombre_original_programa	alfabéticos con acentos, ñ y espacios min 6 max 60
autor_programa	alfabéticos con acentos, ñ y espacios min 6 max 50
autor_original_programa	alfabéticos con acentos, ñ y espacios min 6 max 50
ano_programa	dígitos 4 min 4 max, año no superior actual
ano_original_programa	dígitos 4 min 4 max, año no superior actual
requisitos_programa	alfabéticos con acentos, ñ, espacios y signos de puntuación min 6 max 300
poblacion_desde_programa	dígitos min 1 max 2
poblacion_hasta_programa	dígitos min 1 max 2
unidad_poblacion	solo valores 'MESES','AÑOS'
tipo_programa	solo valores 'EVALUACIÓN' , 'INTERVENCIÓN' , 'EVALUACIÓN E INTERVENCIÓN'
tiempo_aplicacion_programa	dígitos min 1 max 4

descrip_interp_programa	alfabéticos con acentos, ñ, espacios, signos de puntuación y retornos de carro min 100 max 5000
fichero_programa	alfabéticos sin acentos ni ñ ni espacios min 7 max 60. Solo pdf, doc o docx y tamaño de fichero menor de 2000000 bytes.
enlace_programa	alfabéticos y : y / y . sin acentos ni ñ ni espacios min 10 max 100
formato_programa	solo valores ('PAPEL', 'ELECTRÓNICO','PAPEL Y ELECTRÓNICO')
modo_correccion_programa	solo valores ('PAPEL', 'ELECTRÓNICO','PAPEL Y ELECTRÓNICO')
modo_aplicacion_programa	solo valores 'INDIVIDUAL','COLECTIVO','INDIVIDUAL Y COLECTIVO'
imagen_programa	alfabéticos sin acentos ni ñ ni espacios min 7 max 60. Solo jpg o jpeg y tamaño de fichero menor de 20000 bytes.

tabla publicacion

```
CREATE TABLE `publicacion` (
  `id_publicacion` int(6) NOT NULL,AUTOINCREMENTAL
  `titulo_publicacion` varchar(80) NOT NULL,
  `autor_publicacion` varchar(40) NOT NULL,
  `fecha_publicacion` date NOT NULL,
  `categoria_publicacion` int(6) NOT NULL,
  `texto_publicacion` varchar(8000) NOT NULL,
  `imagen_publicacion` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Formatos permitidos

id_publicacion	dígitos min 1 max 6
titulo_publicacion	alfabéticos con acentos, ñ y espacios min 6 max 80
autor_publicacion	alfabéticos con acentos, ñ y espacios min 6 max 40

fecha_publicacion	fecha válida presente o futura con formato dd/mm/aaaa
area_publicacion	dígitos min 1 max 6
texto_publicacion	Cualquier ascii min 30 max 8000
imagen_publicacion	alfabéticos sin acentos ni ñ ni espacios min 7 max 50. Solo jpg o jpeg y tamaño de fichero menor de 20000 bytes.

Objetivos

1) Implementación

Para cada tabla propuesta se solicita la presentación de las tuplas de la tabla con elección de los atributos a mostrar. En la parte superior de la tabla deben estar los iconos para ADD y SEARCH. En cada una de las filas deben estar los iconos para EDIT, DELETE y SHOWCURRENT.

Los iconos deben llevar ante un click a un formulario para realizar la acción. Este formulario se presentará mediante un div con comportamiento modal. Por lo tanto, el formulario debe tener la opción de realizar la acción y la de cancelar la acción. Debe comprobarse cada campo del formulario para verificar su formato (e información si es necesario) en el momento de su introducción si es posible. Debe comprobarse que todos los campos sean correctos antes de enviar al BACK la realización de la acción. Los mensajes de error de campo deben mostrarse de manera que no entorpezcan la interacción del usuario y debe darse una indicación visual en cada campo para mostrar al usuario si es correcto o no su valor.

Los atributos correspondientes con subida de ficheros al BACK tienen un funcionamiento particular:

Aquellos atributos (p.e. xxxx) que se utilizan para almacenar el nombre de los ficheros que se suben al servidor representan el valor del nombre del fichero subido y se muestran con un campo de formulario de tipo input type text que será readonly en el EDIT, DELETE, SHOWCURRENT, no readonly en SEARCH y no se muestra (display none) en ADD.
Se creará un campo de formulario de tipo file para la subida del fichero correspondiente al servidor con id y name “nuevo_xxxx” y se mostrará (display block) en las acciones ADD y EDIT y no se mostrará (display none) en las acciones DELETE, SHOWCURRENT y SEARCH.

También se creará una etiqueta a con id “link_xxxx”, con un contenido correspondiente a un icono de un fichero, para poder colocar un href correspondiente a la dirección del fichero en el servidor. Esta dirección es de la siguiente forma:

http://193.147.87.202/ET2/filesuploaded/files_xxxx/nombredefichero

Ejemplo para el campo fichero_programa de la entidad programa:

<label id=”label_fichero_programa” class=”fichero_programa”></label>
<input type=”text” id=”fichero_programa” name=”fichero_programa”>

<label id=”label_nuevo_fichero_programa” class=”nuevo_fichero_programa”></label>
<input type=”file” id=”nuevo_fichero_programa” name=”nuevo_fichero_programa”>

Si la acción solicitada al BACK es correcta se realizará una actualización de la muestra de las tuplas y quedará a la espera del usuario para una nueva acción desapareciendo el formulario y los mensajes de error. Si la acción solicitada al BACK devuelve un error se indicará al usuario mediante un componente modal para el que usuario confirme que ha recibido el error. Una vez haya confirmado la recepción del error el formulario continuará como antes de enviar la petición a BACK para que el usuario pueda resolver el error indicado.

2) Pruebas

a) Determinar la definición de los test que son necesarios para establecer la validez de los datos introducidos en el formularios y sus mensajes correspondientes, que el usuario debe recibir como respuesta a la introducción de los mismos. Estas definiciones se crearán mediante un array que contenga la entidad, el campo, el número de definición de test, la descripción del test y el mensaje de respuesta que el test debe devolver.

b) Determinar el conjunto de pruebas que se deben realizar para verificar el correcto funcionamiento de los test definidos tanto en su respuesta positiva como negativa.

c) Colocar un icono de test por encima de los botones de ADD y SEARCH para llamar a la función test.

Arquitectura

1) Directorios y ficheros

En la raíz de la web solicitada tendrá un index.html que enviará a una página menu.html en donde estarán disponibles las gestiones de entidades. (Esto lo proporciono yo directamente)

En la raíz de la web solicitada se colocarán todos los ficheros html de las gestiones solicitadas.

Existirá un directorio js_app en donde se colocarán las clases js correspondientes a la gestión de cada entidad (funcionalidad) de la web solicitada. Los ficheros de clases de gestión de entidades tendrán el nombre ‘Gestion_nombreentidad.js’.

Existirá un directorio js_core en donde se colocarán las clases js correspondientes a las funciones comunes de modificación del DOM dentro de un fichero DOM_class.js que contendrá la clase DOM_class en la cual estarán los métodos que se definan para la

manipulación del DOM que se invocarán estáticamente. (NO SE MODIFICA)

Existirá un directorio js_base en donde se colocarán las clases js correspondientes a las funciones comunes js de gestión de funcionalidades. Contendrá por lo menos un fichero GestionEntidad.js con una clase Gestion_Entidad de la cual heredarán todos las clases de Gestion de cada entidad de la base de datos. (NO SE MODIFICA)

En este directorio se colocará el fichero Validaciones_Atomicas.js con la clase validacionesatomicas que contendrá los métodos estáticos correspondientes a las validaciones atómicas de las comprobaciones de los campos de formulario.

Existirá un directorio locales en donde se colocará la información correspondiente al soporte multi idioma de la web solicitada. Se implementará solo el fichero en castellano que llevará por nombre textos_ES.js y la variable en su interior será textos_ES. Esta variable será un array que contenga pares del tipo “codigodetexto”:”valor de texto para ese codigodetexto”. Todos los códigos que se utilicen para el soporte de idioma serán colocados como class en los elementos html en donde se utilicen. La función setLang() será la responsable de modificar sus valores cuando se invoca.

Existirá un directorio con el nombre js_test, que contendrá el fichero tests.js que contiene la funcionalidad para la validación de los ficheros de pruebas. (NO SE MODIFICA) Está subido en github. Esta funcionalidad prueba y muestra en un div el resultado de las pruebas definidas en los ficheros de test solicitados a partir de un botón test que está sobre la tabla de muestra de datos, debajo del título de la página. al pulsar el botón test se muestra un div con los resultados de todas las pruebas definidas. Los ficheros solicitados para las pruebas deben estar también en este directorio, independientemente que estén también en otro lugar en el proceso de entrega.

Existirá un directorio css que contendrá un css para la gestión de elementos modales ya proporcionado por el profesor. (NO SE MODIFICA) De todas formas, si se desea cambiar puede hacerse siempre y cuando siga manteniéndose el carácter modal en formularios y mensajes de errores de acción.

Existirá un directorio iconos que contendrá los iconos utilizados para representar las acciones en la interfaz.

2) Variables y métodos

Los campos en los formularios tendrán el name e id correspondientes a los atributos de la entidad en la tabla.
El nombre e id de los formularios será IU_form.
El nombre e id de los div que contienen los formularios será div_IU_form.
El nombre del método de preparación de los formularios será createForm_*ACCION*()
El nombre del método de comprobación de submit a colocar en el onsubmit será comprobar_submit_*ACCION*()
El formulario tendrá un type submit para provocar el envío del formulario.
El nombre de los métodos de action será la acción a realizar *ACCION*()
El método de comprobación de formato de un campo de formulario será comprobar_*CAMPO*()

3) Interfaz

Todas las acciones estarán representadas por iconos.
Los iconos deben mantener una coherencia visual.

Propósito

1) Realizar la entrega de un un fichero texto ascii con el nombre ET2_NombreApellidosAlumno.js para la indicación de los datos de entrega.

Debe contener el nombre del alumno la entrega, y las horas dedicadas en el total de la entrega con el siguiente formato:

datosgenerales = Array(Apellido1 Apellido2 Nombre (del Alumno), entrega, horasdedicadas);

Ejemplo:

datosgenerales = Array('Rodeiro Iglesias Javier','ET1', 80);

1) Realizar la entrega de un fichero texto ascii con el nombre ET2_NombreApellidosAlumno_tests.js para la definición de los test de formato de campo.

Cada definición de test debe contener la siguiente información:

Número definición test (Consecutivo de 0-número de definiciones totales)
Entidad
Campo
Descripción test
Resultado esperado (true/false)
Mensaje respuesta

```
def_test =
Array(
        //comentario informativo si se desea
        Array(
                Número Descripción test,
                Entidad,
                Campo,
                Descripción Test ,
                Resultado esperado ,
                Mensaje respuesta),
        .
        .
        .
```

);

He aquí un ejemplo:
usuario : alfabético sin acentos ni ñ ni espacios entre 3 y 45 caracteres

Campo	Entidad	Número Definición test	Descripción test	Resultado	Mensaje
usuario	persona	1	Tamaño < 3	false	‘El login del usuario no puede tener menos de 3 caracteres’
usuario	persona	2	Tamaño > 45	false	‘El login del usuario no puede tener más de 45 caracteres’
usuario	persona	3	no alfabético o con acentos o con ñ o con espacios	false	‘Login de usuario contiene caracteres no permitidos (solo alfabéticos sin acentos’)
usuario	persona	4	alfabético sin acentos ni ñ ni espacio	true	‘Exito’

def_test =
Array(
//basicas de usuario
Array('usuario',’persona’,1,’tamaño < 3’ ,false,’El login del usuario no puede tener menos de 3 caracteres’),
Array('usuario',’persona’,2,’tamaño > 45”,false,‘El login del usuario no puede tener más de 45 caracteres’),
Array('usuario',’persona’,3,’no alfabético o con acentos o con ñ o con espacios’,false,’‘Login de usuario contiene caracteres no permitidos (solo alfabéticos sin acentos)’),
Array('usuario',’persona’,4,’alfabético sin acentos ni ñ ni espacios’,true,’Exito’)
);

2) Realizar la entrega de un fichero texto ascii con el nombre ET2_NombreApellidosAlumno_pruebas.js para las pruebas con valores de los tests definidos en el documento anterior.

El fichero tendrá el siguiente formato:

```
pruebasunitarias_ = Array(
   //basicas de usuario
   Array(      número definición de test, (conforme el fichero anterior)
               Entidad,
               campo,
               número test, (consecutivo 0-total de pruebas realizadas para todos los
      campos)
               valor,
               resultado(true/false)
      )
```

He aquí un ejemplo:

Número Definición test	Entidad	campo	número test	valor	resultado
1	persona	usuario	1	jr	false
1	persona	usuario	2	javi	true
2	persona	usuario	3	‘aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaa’	false
2	persona	usuario	4	javi	true
3	persona	usuario	5	j rodeiro	false
3	persona	usuario	6	iñaki	false
3	persona	usuario	7	javier	true
4	persona	usuario	8	jose ford	false
4	persona	usuario	9	jrodeiro	true

```
pruebasunitarias =
Array(
	//basicas de usuario
	Array(1,'persona','usuario',1,'jr',false),
	Array(1,'persona','usuario',2,'javi',true),
	Array(2,'persona','usuario',3,'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaa',false),
	Array(2,'persona','usuario',4,'javi',true),
	Array(3,'persona','usuario',5,'j rodeiro',false),
	Array(3,'persona','usuario',6,'iñaki',false),
	Array(3,'persona','usuario',7,'javier',true),
	Array(4,'persona','usuario',8,'jose ford',false),
	Array(4,'persona','usuario',9,'jrodeiro',true)
);
```

Historias de usuario a cumplir

Particulares de entrega (Obligatorio. Si se incumple alguno de estos criterios la nota será 0)
1. Los ficheros tienen el nombre, formato y tipo indicado en la entrega
2. El directorio a entregar existe y tiene el nombre indicado en la entrega
3. El alumno evaluado ha indicado el número de horas utilizadas en la realización de la entrega

Por cada error en la definición de test a realizar (p.e. el mensaje no es adecuado para la prueba, devuelve un true un test de error, devuelve un false un test de exito, falta el test de exito, falta algun test de error,......) 0,1

Por cada error en la definición de pruebas de test a realizar (p.e., devuelve un true en una prueba de error, devuelve un false una prueba de éxito, falta la prueba de éxito, falta alguna prueba de error,......) 0,1

Por cada error en la construcción del formulario : fallo de comprobación de campo (sintáctico, construcción de nombre, ejecución, …), fallo de comprobación de submit (sintáctico, construcción de nombre, ejecución, …), fallo de obligatorio, no editable, seleccionable….. 0,1

Por cada error en la construcción de la interfaz: falta de iconos, falta de coherencia visual en los iconos, ventanas modales incorrectas, tratamiento de códigos en los ficheros de idiomas, 0,1

Se solicita

1) Los datos de la entrega realizada, identificando la entrega, el alumno y el número de horas dedicadas.

2) El código de FRONT necesario para realizar la gestión de las tablas indicadas estructuradas de la manera que se indica en la sección Arquitectura a seguir. Incluye el código necesario para realizar las pruebas automatizadas de formato.

3) Identificación de los test a realizar por cada campo del formulario con sus mensajes de respuesta.

Se especifica cada tipo de test que debería realizarse para cada campo para detectar los posibles errores de formato junto con el test para detectar el formato válido. **Es importante que las definiciones de test de error devuelvan un false en la prueba para comprobar si el valor es erróneo y devuelvan un true si el valor es válido. Solo la definición del test de formato válido debería responder un true.**

4) Pruebas con valores para cada test y mensajes de respuesta para validación de formato de cada campo.

Forma de entrega

1) En el ejercicio ET2 de moovi de la asignatura se introducirán un fichero ET2_NombreApellidosAlumno.rar. Para ello:

a) Introducir los tres ficheros solicitados en un directorio con nombre ET2_NombreApellidosAlumno.

b) Introducir un directorio “Codigo” con todo el código de FRONT solicitado en el directorio anterior.

c) Comprimir el directorio en formato rar y darle el nombre ET2_NombreApellidosAlumno.rar

(SE ENTREGA ANTES DEL VIERNES DÍA 17 DE NOVIEMBRE A LAS 23:59 HORAS)

Changelog

[1] xx/x/xxxx

- []texto original

por

- []cambio realizado