Definición Entrega ET2
Interfaces de Usuario
Curso 2024-2025

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 `analysis_preparation`
--

CREATE TABLE `analysis_preparation` (
  `id_analysis_preparation` int(11) NOT NULL AUTOINCREMENT,
  `name_analysis_preparation` varchar(100) NOT NULL UNIQUE,
  `description_analysis_preparation` varchar(5000) NOT NULL,
  `bib_analysis_preparation` varchar(200) NOT NULL,
  `file_analysis_preparation` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci;
```

Formatos permitidos

id_analysis_preparation	dígitos min 1 max 11

name_analysis_preparation	alfabéticos y espacios sin acentos ni ñ, min 8 max 100
description_analysis_preparation	alfabéticos y espacios sin acentos ni ñ, min 80 max 5000
bib_analysis_preparation	alfabéticos con acentos, ñ, espacios y signos de puntuación, min 6 max 200
file_analysis_preparation	alfabéticos con punto sin acentos ni ñ ni espacios min 7 max 100. Solo pdf, doc o docx y tamaño de fichero menor de 2000000 bytes.

```
--
-- Estructura de tabla para la tabla `project`
--

CREATE TABLE `project` (
  `id_project` int(11) NOT NULL AUTOINCREMENT,
  `name_project` varchar(500) NOT NULL UNIQUE,
  `start_date_project` date NOT NULL,
  `end_date_project` date NOT NULL,
  `responsable_project` varchar(60) NOT NULL,
  `organization_project` varchar(100) NOT NULL,
  `description_project` varchar(500) NOT NULL,
  `file_project` varchar(100) NOT NULL,
  `code_project` varchar(50) NOT NULL UNIQUE,
  `acronym_project` varchar(15) NOT NULL UNIQUE,
  `id_sampling_methodology` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci;
```

Formatos permitidos

id_project	dígitos min 1 max 11
name_project	alfabéticos y espacios sin acentos ni ñ, min 15 max 100
start_date_project	fecha válida con formato dd/mm/aaaa
end_date_project	fecha válida con formato dd/mm/aaaa y superior a star_date_project
responsable_project	alfabéticos con acentos, ñ y espacios min 6 max 60
organization_project	alfabéticos con acentos, ñ y espacios min

	6 max 100
description_project	Cualquier ascii min 30 max 500
file_project	alfabéticos sin acentos ni ñ ni espacios con punto min 7 max 100. Solo pdf, doc o docx y tamaño de fichero menor de 2000000 bytes.
code_project	alfabéticos con ñ, espacios y signos de puntuación, sin acentos, min 6 max 50
acronym_project	alfabéticos con ñ y signos de puntuación, sin acentos ni espacios, min 6 max 15
id_sampling_methodology	dígitos min 1 max 11

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 un array en momento de inicialización de la clase y un elemento de selección múltiple en la parte superior derecha de la tabla. 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) el mismo ni su label (label_xxxx) 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 un label con id label_nuevo_xxxx y se mostrarán (display block) en las acciones ADD y EDIT y no se mostrarán (display none) en las acciones DELETE, SHOWCURRENT y SEARCH.

También se creará una etiqueta a con id “link_xxxx”, con un contenido correspondiente a una imagen de 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.
La definición de tests, las pruebas de campo, y las pruebas de campo file de cada entidad estarán en un fichero con nombre tests_nombreentidad.js en el directorio js_app.

Estas definiciones se crearán, para los campos del formulario, mediante un array de nombre def_tests_nombreentidad que contenga:

la entidad,
el campo,
el número de definición de test (secuencial desde 1 hasta el final)
la descripción del test
la acción a realizar
el resultado esperado para este test (boolean/string)
el mensaje de respuesta asociado al resultado.

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. Estas pruebas se crearán, para los campos del formulario que no sean input tipo file, mediante un array de nombre ‘pruebas_nombreentidad’ que contenga:

la entidad,

el campo,
el número de definición de test,
el número de prueba (secuencial desde 1 hasta el final)
la acción a realizar
el valor a probar
el código asociado de error/valor true de éxito

Para los campos de formulario input de tipo file, se usará el siguiente array de nombre 'pruebas_file_nombrentidad'

la entidad,
el campo,
el número de definición de test,
el número de prueba (secuencial desde 1 hasta el final)
la acción a realizar
el parámetro a probar (max_size_file, type_file, format_name_file)
el valor de parámetro a probar
el codigo asociado de error/valor true de exito

c) Existe en la interfaz un icono de test en el header del index.html para llamar a la función test.

Arquitectura

1) Directorios y ficheros

En la raíz de la web solicitada tendrá un index.html en donde estarán disponibles las gestiones de entidades a través de un menú. (Proporcionado y se usa sin modificaciones)

Existirá un directorio js_app en donde se colocarán las clases js correspondientes a la gestión de cada entidad de la web solicitada. Los ficheros de clases de gestión de entidades tendrán el nombre 'nombreentidad.js'.

También estará dentro de js_app el fichero Validaciones_Atomicas.js con la clase validacionesatomicas que contendrá los métodos correspondientes a las validaciones atómicas min_size(id, parametro), max_size(id,parametro) y format(id,parametro) de las comprobaciones de los campos de formulario y max_size_file(objfile, parametro), type_file(objfile, array de tipos permitidos), format_name_file(objfile, parametro) de los campos file.

También estará dentro de js_app el fichero Tests_class.js que contiene la clase test para la validación de los ficheros de pruebas. 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 al lado del título de la página y que llama al metodo test_run(). 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 el directorio js_app.

Dentro del directorio js_app se colocará un fichero ET2_datosgenerales.js tal y como se indica en la sección Propósito.

Existirá un directorio js_core en donde se colocará la clase 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 (Proporcionado y se usa sin modificaciones). En este directorio también se encontrará el fichero ExternalAccess.js con la clase ExternalAccess para el acceso a Back (Proporcionado y se usa sin modificaciones).

Existirá un directorio js_base en donde se colocarán la clase js con los métodos comunes de gestión de entidades. Contendrá un fichero EntidadAbstracta.js con una clase EntidadAbstracta de la cual heredarán todos las clases de gestión de cada entidad. (Proporcionado y se usa sin modificaciones)

Existirá un directorio locale en donde se colocará el proceso correspondiente al soporte multi idioma de la web solicitada.

En el directorio js_app se implementará 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. De la misma forma se implementará el fichero de soporte en inglés que llevará por nombre Textos_EN.js y la variable interior será textos_EN.

Existirá un directorio css que contendrá un css para la gestión de elementos modales ya proporcionado por el profesor. (Proporcionado y se usa sin modificaciones). 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 y se implementará creando un fichero con nombre “micss.css” que se colocará en el directorio js_app y se indicará su uso en un fichero README.txt también en js_app.

Existirá un directorio iconos que contendrá los iconos utilizados para representar las acciones en la interfaz. (Proporcionado y se usa sin modificaciones)

2) Variables y métodos

a) ids de objetos DOM

- Los campos en los formularios tendrán el name e id correspondientes a los atributos de la entidad en la tabla. Además se tendrá en cuenta los ids indicados para los ficheros a subir.

- Los divs y formulario a usar en la entrega corresponden con los que están en la página index.html.

- Existe un array definido a nivel de la clase de entidad denominado columnasamostrar con los nombres de los atributos que se quieren mostrar en la tabla de presentación de tuplas.

- Existe un array definido a nivel de la clase de entidad denominado datosespecialestabla con los nombres de los atributos que de los cuales se quieren modificar su valor a la hora de mostrarlo en la tabla de presentación de tuplas para no hacerlo por defecto contra su valor en la tupla.

- validacion especial (se invoca a validacionesespeciales(id,prueba) en las comprobaciones de campo. No se colocan en validaciones, sino en la clase de entidad concreta donde se usan

b) Métodos de clases

- 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 método de comprobación de formato de un campo de formulario será comprobar_*CAMPO*_ACCION(). Los métodos de comprobación de campo deben responder true si es correcto y el código de error si es incorrecto.

Como mínimo para acción se ha de distinguir entre SEARCH y las demás. Si se necesita diferenciar en comprobar_atributo() entre el ADD y el EDIT se puede utilizar, como está implementado en el ejemplo de persona, un atributo de acción que se crea solo con ese fin si es necesario.

- El formulario tendrá un botón de tipo submit con un icono para provocar el envío del formulario.

- los métodos estándar en la clase validacionesatomicas son min_size(), max_size(). format(), max_size_file(), type_file(), format_name_file().

- el método para las validaciones especiales se llama validacionesespeciales(atributo, prueba) y se utiliza para codificar todas las validaciones que no corresponden con los métodos estándar de la clase validacionesatomicas.

- Si se definen atributos en datosespecialestabla deberá existir un método cambiardatosespecialestabla(atributo, valoratributo) en la tabla de la entidad en donde para cada atributo definido en datosespecialestabla debe existir la devolución de la información a mostrar para ese atributo.

- la modificación de valores del formulario se realizará en el metodo createForm correspondiente a la acción.

- El nombre de los métodos de action será la acción a realizar *ACCION*()

3) Interfaz

Todas las acciones estarán representadas por iconos. Los iconos son los proporcionados en el directorio iconos

Propósito

1) Realizar la entrega de un fichero texto ascii con el nombre ET2_datosgenerales.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);

2) Realizar la entrega de un directorio js_app con el código solicitado desarrollado en los ficheros solicitados y codificados como se indica en la definición de la entrega.

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

Por cada error en ejecución de los test de entidad: 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 tal y como se indica en la definición

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

Forma de entrega

1) Crear un directorio con el nombre ET2_NombreApellidosAlumno.

2) Introducir el directorio “js_app”, con todo el código de FRONT y ficheros solicitados y desarrollados por el alumno, en el directorio ET2_NombreApellidosAlumno.

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

4) Entregar en la tarea Entrega ET2 de moovi

(SE ENTREGA ANTES DEL VIERNES DÍA 22 DE NOVIEMBRE A LAS 23:59 HORAS)