# Definición Entrega ET2
# Interfaces de Usuario
# Curso 2025-2026

## 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

```
CREATE TABLE `articulo` (
  `CodigoA` int(11) NOT NULL COMMENT 'Código Artículo',
  `AutoresA` varchar(200) NOT NULL DEFAULT '' COMMENT 'Nombres autores',
  `TituloA` varchar(100) NOT NULL DEFAULT '' COMMENT 'Título Artículo',
  `TituloR` varchar(100) NOT NULL DEFAULT '' COMMENT 'Título Revista',
  `ISSN` varchar(13) NOT NULL DEFAULT '' COMMENT 'ISSN',
  `VolumenR` varchar(4) NOT NULL COMMENT 'Volumen de la revista',
  `PagIniA` int(4) NOT NULL DEFAULT 0 COMMENT 'Número página Inicial artículo',
  `PagFinA` int(4) NOT NULL DEFAULT 0 COMMENT 'Número página final artículo',
    `FechaPublicacionR` date NOT NULL DEFAULT '0000-00-00' COMMENT 'Fecha publicación artículo',
  `FicheropdfA` varchar(20) NOT NULL COMMENT 'fichero artículo',
    `EstadoA` enum('Enviado','Revision','Publicado') NOT NULL DEFAULT 'Publicado' COMMENT 'Estado de tramitación del artículo'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `ubicacion` (
  `id_site` int(11) NOT NULL COMMENT 'Identificador Sitio',
  `site_latitud` decimal(9,6) NOT NULL COMMENT 'Latitud WGS84',
  `site_longitud` decimal(9,6) NOT NULL COMMENT 'Longitud WGS84',
  `site_altitude` int(4) NOT NULL COMMENT 'Altitud Sitio',
  `site_locality` varchar(40) NOT NULL COMMENT 'Localidad del sitio',
  `site_provider_login` varchar(30) NOT NULL COMMENT 'Login proveedor sitio',
  `site_north_photo` varchar(50) NOT NULL COMMENT 'foto hacia el norte',
  `site_south_photo` varchar(50) NOT NULL COMMENT 'foto hacia el sur',
  `site_east_photo` varchar(50) NOT NULL COMMENT 'foto hacia el este',
  `site_west_photo` varchar(50) NOT NULL COMMENT 'foto hacia el oeste'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

## Objetivos

1) Implementación

Para cada tabla propuesta y la de alumnograduacion de la ET1, 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”>
<a id=”link_fichero_programa” href=””><img src=”nombreficheroiconofichero.jpg” /></a>
<br>
<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 ET2_infotest.js en el directorio app.

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

la entidad,
el campo,
elemento de formulario que usa (‘input’, ‘inputfile’, ‘select’, ‘checkbox’,’radio’, ‘textarea’)
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 ‘nombreentidad_test_fields’ 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 'nombrentidad_test_files'

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 (not_exist_file, max_size_file, type_file, format_name_file)
el valor de parámetro a probar
el codigo asociado de error/valor true de éxito

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

## 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 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_Class.js' y la clase el nombre de la entidad.

Dentro de core está el fichero Validations_Class.js con la clase Validations que contendrá los métodos correspondientes a las validaciones estándar min_size(id, parametro), max_size(id,parametro) y format(id,parametro) de las comprobaciones de los campos de formulario y not_exist_file(id), max_size_file(id, parametro), type_file(id, array de tipos permitidos), format_name_file(id, parametro) de los campos file. (Proporcionado y se usa sin modificaciones)

Dentro de core estará el directorio Test que contiene las clases para la validación de los test y pruebas definidas. Esta funcionalidad 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. Al pulsar el botón test se muestra un div con los resultados de todas las pruebas definidas. (Proporcionado y se usa sin modificaciones)

Los ficheros solicitados para las pruebas deben estar en el directorio app.

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

Dentro del directorio app se colocará un fichero formatosPermitidos.txt en el cual se indicará para cada atributo de cada entidad el formato permitido de la misma manera en el

que se indicó para la entidad en la ET1.

Existirá un directorio Core en donde se colocará la clase js correspondientes a las funciones comunes de modificación del DOM dentro de un fichero Dom_class.js y Dom_Table_Class.js que contendrá la clase dom y dom_table en la cual estarán los métodos que se definan para la manipulación del DOM (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 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. (Proporcionado y se usa sin modificaciones)

En el directorio 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).

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 y usados en el repositorio en la semana 6.

- 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 mostrarespecial 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. Si tenemos atributos en ese array se implementará un método mostrarcambioatributo(atributo, valorentrada) que devuelve el formato especial a mostrar del valor del atributo habiéndose pasado el nombre del atributo y su valor a modificar.

- Si se implementan métodos para validaciones porque no valen los que están en validaciones empezarán siempre por “personalize_atributo”.

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 *ACCION*_submit_*ENTIDAD*(). El método de comprobación de formato de un campo de formulario será *ACCION*_campo_validation(). Los métodos de comprobación de campo deben responder true si es correcto y el código de error si es incorrecto.

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

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

- la modificación de valores del formulario se realizará en el método 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 dentro del directio app.

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

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

2) Realizar la entrega de un fichero formatosPermitidos.txt con el formato permitido de

cada campo de formulario en el directorio app.

3) Realizar la entrega de un directorio app con el código solicitado desarrollado en los ficheros solicitados y codificados como se indica en la definición de la entrega además del resto de ficheros solicitados.

## 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.

4) Los ficheros de traducciones para español e inglés tal y como se han indicado.

5) Fichero de descripción de formatos válidos para los atributos

## Forma de entrega

1) Crear un directorio con el nombre ET2_NombreApellidosAlumno.

2) Introducir el directorio “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 14 DE NOVIEMBRE A LAS 23:59 HORAS)