Deusto
Facultad de Ingeniería
Ingeniaritza Fakultatea

Proyecto web colaborativo: E3 + E4

Asignatura	Ingeniería Web
Curso	Curso 2024/2025
Fecha	07/05/2025
Tipo de entrega	Grupos de 2 y 3 personas * En caso de no poneros de acuerdo, se harán los grupos de forma aleatoria.
Fecha límite de entregas	• E3+E4 → 28/05/2025 a las 12:00

1. Desarrollo técnico (E3)

A continuación se detallan las funcionalidades a desarrollar en el proyecto, así como su puntuación:

- **(4 puntos):** Ampliación de funcionalidades en Python. Escoge las funcionalidades que quieras de las listadas a continuación (máximo si podrán lograr 4 puntos):
  - (2 puntos) Envío de emails desde la aplicación
  - (2 puntos) Subida de ficheros al servidor mediante <input type="file"> y mostrarlos en una página (tienen que poder descargarse)
  - (2 puntos) Autenticación y registro de usuarios. La aplicación no mostrará las páginas si el usuario no se ha autenticado previamente.
  - (2 puntos) Buscador de registros, ofreciendo al usuario al menos la posibilidad de filtrar por dos campos distintos.
  - (1 punto) Buscador simple de registros por un único campo.
  - (1 punto) Registro de mensajes (informativos, avisos, errores) mediante Logger
  - (1 punto) Paginación en tablas/listados de los resultados de una tabla.

  * Se trata de un apartado de investigación/ampliación. El objetivo es, partiendo de los conocimientos adquiridos en clase, demostrar la capacidad de ampliar vuestros conocimientos trabajando en equipo.
  * Si las funcionalidades mínimas requeridas en la E2 no se encuentran funcionando correctamente, la puntuación de este apartado será de 0.
  * En caso de implementar más de dos funcionalidades, se tendrá en cuenta de forma positiva (pero no se podrán obtener más de 4 puntos en este apartado).

- **(3 puntos)** Implementar las siguientes funcionalidades JavaScript (escoger 3 de ellas, en caso de implementar más de 3 opciones se tendrán en cuenta las 3 con mejor puntuación):
    - (1p) Crear un evento al hacer click en un botón o enlace que produzca el cambio (aumentar/disminuir) del tamaño de todos los elementos de texto (<h1>, <h2>, <p>, ...).
    - (1p) Validar campos de un formulario antes de su envío al servidor, impidiendo el envío si no se supera la validación (p. ej: formato de DNI correcto, comprobar que se ha introducido el mismo valor en dos campos, permitir únicamente contraseñas que contienen ciertos caracteres, impedir que se introduzca un nombre de usuario perteneciente a una lista de palabras reservadas, ...). Mostrar el mensaje de error tras la validación.
        - Las validaciones tienen que aportar valor, es decir, no se contabilizarán validaciones que ya se pueden hacer de forma simple mediante HTML (campo vacío o required, email válido, campo númerico dentro de un rango, etc.).
    - (1p) Autocalcular un campo de un formulario. Por ejemplo: generar el campo de username a partir del nombre y apellidos del usuario (Jon Vadillo → jon.vadillo), hacer una operación matemática con el valor de dos campos, seleccionar automáticamente el valor de un campo en función de otra selección previa, convertir los caracteres seleccionados a mayúsculas,...)
    - (1p) Generar contenido HTML a partir de un array de datos (el array puede incluir objetos, strings, ...) y añadirlo al DOM (p. ej: incluir los valores de un elemento de tipo <select>, crear un menú generando cada elemento a partir de los valores de un array, ...).
    - (1p) Capturar un evento en el DOM y producir un cambio en el estilo/contenido de la página (p. ej: mostrar/ocultar un bloque al hacer click en “expandir información”, mostrar una alerta si el usuario realiza una acción determinada,...)

  * Para aprobar la entrega es necesario demostrar que habéis adquirido un mínimo de conocimientos de JS, por lo que **se requiere obtener la mitad de los puntos (1’5 puntos) de esta parte para aprobar la entrega** .
  ** Es posible proponer la implementación de otras funcionalidades que impliquen captura de eventos y modificación del DOM. En caso de querer proponer alguna que no figure en los puntos anteriores, tendréis que escribir un email explicando la propuesta **antes del 19/05/2025 a las 14:00**.

- **(2 puntos)** Funcionalidades JavaScript para cargar y/o almacenar datos utilizando Fetch. Escoger una de las siguientes opciones:
    - Cargar datos y modificar el DOM mediante JavaScript: llamada a API de Django utilizando Fetch para obtener datos y mostrar los valores modificando el DOM (p. ej: ampliar los detalles de un registro que se muestran al usuario).
    - Envío de datos de un formulario mediante AJAX para su almacenamiento en BBDD y posterior visualización del resultado (p. ej: cambiar el estado de un

registro de “en proceso” a “completado” y mostrarlo en pantalla sin necesidad de recarga de la página).

- **(1 punto)** Calidad del código entregado (estructura correcta, uso de las técnicas adecuadas que se han visto en clase, limpieza, comentarios en el código, ausencia de errores en la sintaxis o durante la ejecución de la aplicación,...)
- **Penalizaciones**
    - (1 punto) El número de commits realizados es mínimo o no se han realizado commits de forma regular (deben figurar commits todas las semanas no vacacionales) y por todos los integrantes del equipo.
    - (2 puntos) El código no es limpio, no está bien formateado u ordenado, la implementación es muy mejorable (por ejemplo, unas plantillas tienen herencia y otras no, hay código repetido que podría reutilizarse mediante funciones, apenas se utilizan Vistas Basadas en Clases, al editar campos de un formulario se pierden valores,…).
    - (2 puntos) El código contiene errores de sintaxis, de ejecución (de Python o JavaScript, aunque sea en la consola), no se controla si saltan errores al dejar vacío algún campo, ...

Para obtener la puntuación completa de las funcionalidades de Javascript, se tendrá en cuenta la calidad de la solución aportada. Ejemplos:

- La funcionalidad se consigue pero el código no está organizado en funciones, realiza acciones innecesarias o mejorables, etc.
- La funcionalidad se consigue pero de forma muy mejorable (por ejemplo: se hace una validación y el resultado sale en un “alert” en lugar de construir un mensaje en HTML y con estilos CSS dentro de la página).
- La funcionalidad implementada es muy básica o no tiene sentido (al hacer un click, mostrar un contenido que no aporta nada, o su contenido es muy básico, desentona con los estilos de la página,...
    - Ejemplo de funcionalidad sin sentido: comprobar que un campo de texto está vacío → no tiene sentido porque se puede conseguir añadiendo “required”.

En caso de que el proyecto no se pueda cargar correctamente (mediante el comando *runserver*) y probar, la calificación será automáticamente cero.

2. Presentación grupal del proyecto (E4)

En la segunda fase deberá realizarse una presentación/demo que incluya los siguientes contenidos:

1. **Presentación** y explicación breve del proyecto (45’’).
2. **Diseño** (2’): funcionalidades planteadas, modelo de datos diseñado y descripción de las interfaces
3. **Planificación y organizació**n en equipo (1.5’)

4. **Implementación/demo** (3’-4’): es la parte principal del proyecto, en la que se explicará técnicamente cómo se han realizado las distintas funcionalidades y se mostrarán funcionando correctamente. Es la parte donde se demostrará especialmente que se tiene un entendimiento y dominio del código desarrollado, por lo que tendrán que participar al menos 2 integrantes del equipo.
5. **Conclusiones**.(45’’)

Como se entregará:

- Diapositivas en formato PDF y Powerpoint (se entregará en los 2 formatos).
- Vídeo de **máximo 10 minutos** de duración (si se excede la duración, la calificación de la E4 será de cero):
    - La participación de los miembros del equipo tendrá que ser equilibrada.
    - Durante el vídeo se podrá alternar entre la presentación y la aplicación desarrollada, mostrando claramente las funcionalidades desarrolladas en funcionamiento y con datos suficientes para simular una situación real.
    - La imagen de la webcam con la persona que habla deberá verse en todo momento.
    - Algunas opciones de grabación: Loom, OBS, Screencastify,...

3. Competencia genérica (E4)

CG5. Integrarse en equipos de trabajo y **colaborar de forma activa con otras personas**, áreas y organizaciones en la consecución de los objetivos ligados a las actividades de digitalización industrial.

Se deberá utilizar **Github Projects** y su tablero de tareas para la coordinación de tareas durante el desarrollo del proyecto. Para la entrega se deberán realizar 3 pantallazos del tablero de tareas en al menos 3 momentos:

1. Al comienzo, tras definir todas las tareas.
2. Cualquier día de la semana del 19/05-25/05 (a mitad de desarrollo del proyecto)
3. Al finalizar el proyecto (no importa si hay tareas sin finalizar).

La siguiente rúbrica de evaluación puede utilizarse como referencia a la hora de conocer la importancia de cada criterio.

Criterio	Peso	Excelente	Bueno	Aceptable	Insuficiente
Distribución de commits entre miembros	**10%**	Todos los miembros realizan al menos el 20% del total de commits y ningún	Todos participan, pero hay desequilibrios: un miembro supera el 50% o uno	Solo 2 miembros participan activamente (>85% de los commits); el	Un único miembro realiza la mayoría de los commits

		miembro supera el 50%.	contribuye menos del 15%.	tercero contribuye poco (<15%).	(>80%).
Mensajes de commit claros y significativos	**10%**	Los commits contienen mensajes descriptivos y consistentes. Uso de convenciones (por ejemplo, feat:, fix:) por parte de todos.	Mayoría de los mensajes son comprensibles, pero sin una convención clara o con algunos mensajes genéricos.	Varios commits con mensajes vagos o irrelevantes (por ejemplo, "update", "cambios").	La mayoría de los mensajes de commit son inservibles o vacíos.
Actividad constante durante el proyecto (no trabajo de último momento)	**10%**	Actividad distribuida de forma regular durante el proyecto. Commits y PRs repartidos en el tiempo.	Algún pico de actividad, pero con trabajo en varias semanas.	Trabajo concentrado en una sola semana o en pocos días.	Todo el trabajo ocurre en los últimos días antes de la entrega.
Uso de ramas (branches) por parte de los miembros	**10%**	Todos los miembros han creado y trabajado en ramas propias, con integración mediante pull requests.	Al menos dos miembros usan ramas de forma activa, el tercero colabora desde la rama principal o contribuye en menor medida.	Solo un miembro crea y usa ramas, el resto trabaja directamente en main.	No se utilizan ramas. Todo el trabajo se hace en main o en una sola rama sin control colaborativo.
Uso de Github Projects	**40%**	El equipo utiliza activamente GitHub Projects en modo Kanban. Cada tarea está asociada a un issue con una prioridad clara (label), persona asignada, descripción clara, estado actualizado (To Do / In Progress / Done). Todas las tareas se mueven correctamente por el tablero y se cierran al finalizar. Todos los miembros del equipo participan	El tablero Kanban se usa de forma funcional: hay tareas con asignaciones y alguna priorización. Algunas tareas no tienen label o están mal clasificadas, pero el flujo general se mantiene. Se cierran la mayoría de las tareas y todos los miembros del equipo participan en la gestión de tareas.	Se ha creado un Project, pero el uso es limitado. Pocas tareas tienen responsables o prioridades. El movimiento entre columnas es esporádico y no refleja fielmente el estado del trabajo, o solo es coordinador por uno o dos miembros del equipo.	No se utiliza GitHub Projects, o solo hay una columna sin organización ni seguimiento del trabajo. No hay tareas asignadas, ni flujo visible.

		de manera equitativa.			
Presentación del proyecto	**20%**	Todos los miembros participan de forma equilibrada en la presentación. Explican con claridad el flujo de trabajo, las decisiones técnicas y la organización del equipo (uso de ramas, issues, Project, roles). Demuestran comprensión global del proyecto y coordinación.	Todos los miembros participan, aunque con cierto desequilibrio (alguno habla claramente más). La planificación y organización se explican correctamente, aunque con menor profundidad o claridad.	Solo dos miembros participan activamente. La explicación de la planificación es parcial o confusa. Se percibe falta de preparación conjunta.	Alguna de las personas no presenta o la participación es simbólica. No se explican adecuadamente los aspectos organizativos ni la planificación. Falta cohesión en el discurso.

Por otro lado, se entregará un documento donde se detalle los trabajos realizados por cada miembro del equipo, en orden cronológico.Este documento deberá incluir las tareas realizadas desde el inicio del proyecto (E2).

Título de tarea y breve descripción	Responsable	Fin	Inicio
Crear los modelos de las entidades principales Producto y Categoría.			
Desarrollar la vista y la plantilla para mostrar la lista de Productos			

* Entregar en formato Excel

4. Normas de entrega (E3+E4)

Normas de entrega:

- **Fecha y hora límite: 28/05/2025 a las 12:00. <u>No se admitirán entregas posteriores y no se realizarán excepciones (una entrega posterior se calificará con un cero)</u>.**
- La entrega solo podrá realizarse en grupo, **no se aceptarán entregas individuales**. Si un miembro del equipo queda excluido del mismo por causa justificada y con conocimiento del docente (por ejemplo porque el resto de miembros del equipo reportan una clara falta de actividad/implicación durante el desarrollo del proyecto), verá su entrega suspendida.
- Entrega en ALUD:

    - Al margen del trabajo en el repositorio, también habrá que subir la entrega a ALUD.
    - Cread un archivo comprimido (se admiten ZIP, GZIP o TGZ, RAR) con el prefijo IW seguido del nombre del equipo (EQUIPO1, EQUIPO2, EQUIPO3, ...) y del sufijo E3E4, por ejemplo: IW-EQUIPO1-E3E4.zip (o en lugar de ".zip" la extensión del compresor que hayáis utilizado). **No cumplir con la nomenclatura supondrá una penalización de 1 punto**.
    - El archivo comprimido debe contener:
        - El proyecto de Django al completo (incluida la base de datos)
        - Una imagen del esquema de la BBDD
        - Un archivo .txt con el contenido del comando “pip freeze” y un archivo README.md con la siguiente información:
            - Enlace al repositorio de Github. Si el repositorio no fuese público, será necesario dar permisos de colaborador al usuario “jvadillo”.
            - Puede contener documentación adicional de ayuda o que consideréis relevante (por ejemplo, en caso de alguna funcionalidad que no se haya conseguido al 100%, forma de probar alguna funcionalidad concreta, etc.).
        - El documento de organización y planificación de tareas
        - Al menos 3 pantallazos del tablero de tareas del proyecto Github: una de la primera semana con todas las tareas definidas, otra de la segunda semana y una última al finalizar el proyecto. El nombre de cada imagen debe ser el de la fecha (por ejemplo: 19-05-2025.JPG).
        - Diapositivas de la presentación (en formato Powerpoint y PDF) y archivo del vídeo.
    - NO deberá incluirse el entorno virtual.
- A la hora de comprimir el proyecto para entregarlo, el proyecto deberá contener al menos 8 registros de cada entidad en la base de datos. Por ejemplo:
    - Para el reto uno, se entregará con al menos 8 equipos y 8 tickets.
    - Para el reto dos, se entregará con al menos 8 pedidos y 8 productos.
- Prestad atención a los requerimientos y los indicadores, por favor, y a la fecha límite de entrega.
- Al margen de la entrega comentada, **cada equipo tendrá que completar el siguiente formulario (enlace)**, donde se indican las funcionalidades implementadas en cada apartado.