Aulas virtuales - Mamás 2.0

Vamos a realizar una aplicación WEB que integre las siguientes tecnologías:

- **PHP**. Programación sin *framework* orientada a **MVC**.
- Integración de ***reCaptcha*** versión 3.
- Generación de elementos del ***DOM*** con JavaScript nativo y con ***jQuery***.
- Gestión de **eventos** con JavaScript nativo y con **jQuery**.
- Gestión de código usando **Git** y **GitFlow**.
- Despliegue basado en **contenedores**.
- MDBootstrap

Nuestra aplicación tendrá los siguientes roles: ***administrador***, ***alumno*** y ***profesor***.

Administrador

El **administrador** gestionará: altas, bajas, modificaciones, activaciones y accesos de los usuarios. El administrador será otro profesor que podrá ir cambiando de rol en cualquier momento de su sesión.

Profesor

El **profesor** podrá realizar exámenes que constarán de una pregunta y respuestas: texto, opciones varias, opciones excluyentes y números.
Tendrá las siguientes funcionalidades: **crear examen**, **activar/desactivar examen**, **corregir examen**.

Crear examen: Podrá crear un examen desde cero, añadiendo preguntas de los tipos indicados y las respuestas. Cuando sea texto y número es trivial. Cuando sean opciones se guardarán las opciones posibles y la/s respuesta/s correcta/s en cada caso.
Sería una funcionalidad extra muy bien valorada que un profesor pudiera reutilizar, en algún momento, la pregunta de otro examen. Que exista alguna opción importar o ver preguntas del histórico que tenga y que pueda incorporarlas al examen que esté realizando en ese momento.

Activar/Desactivar: Se podrá poner una fecha y hora de activación/desactivación y/o activar/desactivar manualmente el examen.
Corregir examen: Podrá tener las dos siguientes opciones: corrección automática que realizará precisamente eso, corregirá automáticamente los exámenes de los alumnos y les pondrá una nota final (los campos texto se pueden dejar sin corregir, como optativo se podría realizar una corrección por palabras claves). La otra opción es revisión que permitirá al profesor ir revisando examen por examen y podrá poner notas manuales a las preguntas (por ejemplo para corregir las de texto). Una vez que un examen se ha dado por corregido los alumnos podrán ver sus notas y la corrección del mismo.

Alumno

Sus funcionalidades serán: **realizar examen** y **ver notas**. En su pantalla principal deberían aparecer los exámenes activos en ese momento que todavía no ha realizado. Mientras esté activo el examen podrá realizarlo las veces que quiera, guardándose la última vez únicamente. Las preguntas de tipo opción se realizarán utilizando la técnica de ***drag and drop***; aparecerá la pregunta y debajo dos áreas: en una estarán las opciones con las que puede responder que serán arrastradas al área donde estarán las respuestas que el alumno cree correctas.
Finalmente, también podrá ver su evolución académica revisando las notas de todos sus exámenes.
Todos los usuarios tendrán las opciones comunes de modificación de perfil y pérdida de contraseña.

Consideraciones generales

Respecto a la **planificación** y **otras consideraciones**:

- Se valorará una **planificación de tareas** basada en funcionalidades concretas, realistas y bien estimadas en dificultad y/o duración.
- Una **base de datos** correctamente diseñada e implementada.
- El uso de **repositorios** y ramas de desarrollo.
- **Tableros** bien organizados y actualizados.
- **Documentación** inicial y final útil, completa (que no tiene por qué ser demasiado extensa) y funcional. Esta documentación incluirá al menos:
    - Estudio de los usuarios.

- Historias de usuario.
- Mapa de historias de usuario
- Wireframes
- Guia de estilos.
- Reseña de las decisiones de diseño. Propias, no las del framework.
- Pantallas descargadas.
- Modificaciones realizadas al diseño básico del framework.

- **Se realizará por parejas.** Se indicará claramente qué tarea ha ido realizando cada integrante del grupo y se documentará en el código las partes del mismo que ha realizado cada uno.

Respecto a **PHP**, se valorará:

- Programación basada en **MVC orientada a objetos.**
- Prevención de **inyección de SQL** en las consultas.
- **Encriptación** de la contraseña en la base de datos.
- **Programación robusta**, con control de errores y fichero de bitácora.

Respecto a **JS,** se valorará:

- Programación orientada a objetos.
- Código cuidado, optimizado, bien estructurado, legible.
- Validación correcta de formularios y gestión de los eventos correspondientes.
- Gestión de eventos **drag and drop** en las preguntas de tipo opción.
- Integración de reCaptcha v3 de Google utilizando php.
- Generación de elementos del **DOM** utilizando **JS nativo y jQuery**.

Respecto a Git, se valorará:

- Correcto uso e implementación de **Git** y **GitFlow**.
- Correcto **manejo de las ramas, funcionalidades y etiquetado del código**.
- Correcto **uso de Releases**.
- Manejo de **Pull Request y código compartido**.
- Visualización del estado del **proyecto en GitHub**.

Respecto a Contenedores, se valorará:

- Implementación de los **contenedores personalizados al problema.**
- Implementación de la **interrelación entre los distintos contenedores** para el problema indicado.
- Pruebas y **despliegue de al menos dos alternativas de solución** usando distintos servidores de página web.
- Análisis e **implementación de la mejor manera para la persistencia de la información** dado el problema.
- **Optimización** de los contenedores.

- **Publicación de los contenedores en Docker Hub**.

Respecto de la interfaz se valorará:

- Diseño responsive. Dos diseños: PC y movil.
- Correspondencia entre Wireframes y diseño final
- Estética.
- Decisiones de diseño propias (no las del framework)
- Aportaciones al diseño que hagan la interfaz más usable o atractiva (será igualmente negativo lo contrario)
- Modificaciones de la interfaz por defecto del framework.