Documentación de aplicación web: Gestor de usuarios y Tareas

Carlos López

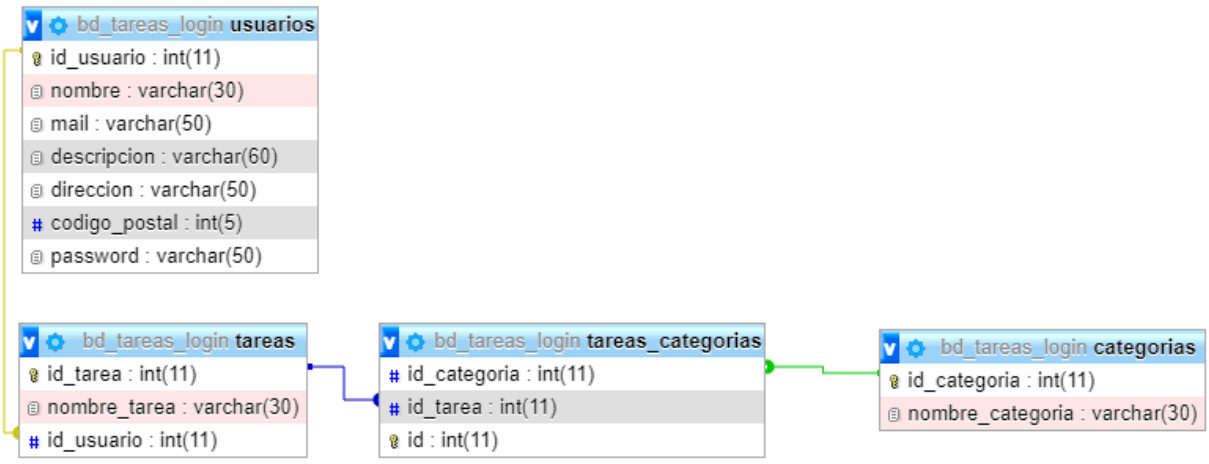
Contenido

Pasos realizados para el desarrollo del proyecto.

Estructura de la base de datos.

En primer lugar, se han creado las entidades correspondientes para que las tareas y los usuarios se puedan organizar debidamente.

La estructura a seguir ha sido la siguiente:



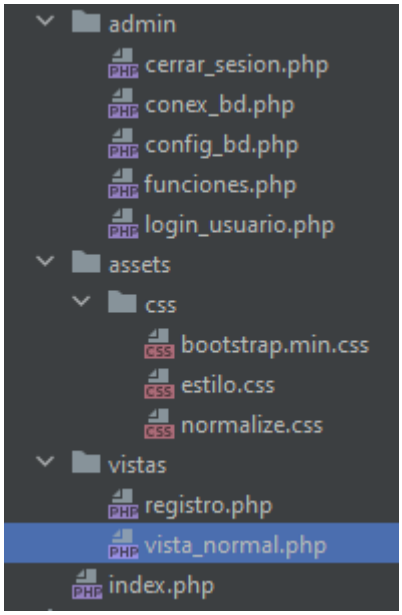
Relación usuarios/tareas: Esta relación se utiliza tanto como para crear tareas que pertenezcan al usuario correspondiente, como para poder listar a cada usuario logueado únicamente las tareas que le pertenezcan.

Relación tareas/categorías: Ya que esta relación es de muchos a muchos (ManyToMany), necesitamos una tabla auxiliar entre las dos entidades, para poder asignar varias categorías a una misma tarea, o que una categoría pueda asignarse a más de una tarea distinta.

Pasos realizados para realizar el código.

Estructura de directorios y ficheros.

En primer lugar, se ha generado la estructura del proyecto, la cual es la siguiente:



En la carpeta admin, se encuentran los ficheros con los que vamos a gestionar la parte de backend:

- Funciones para crear/listar/eliminar tareas del usuario y comprobar si un usuario o tarea ya existe a la hora de crearlos nuevos.
- La conexión a la base de datos.
- La configuración de credenciales para conectar con la base de datos.
- El login para poder ingresar a la aplicación.
- El logout para cerrar la sesión actual.

En la carpeta de assets, se encuentran los ficheros con los que se gestiona la parte de frontend:

- Carpeta css: Contiene las hojas de estilos con las que daremos forma a los diferentes elementos de la aplicación (formularios, botones, etiquetas…).
- Carpeta vistas e Index.php: En estos ficheros se encuentran las vistas que vamos a necesitar para poder realizar todas las funciones del backend (registro, login, gestión de tareas…).

Test de la aplicación.

Una vez finalizado el desarrollo de la aplicación, se han realizado diferentes casos de prueba para comprobar que la misma funciona correctamente.

Los diferentes casos de prueba que se han realizado son los siguientes:

- Comprobación de registro:
  - Se ha comprobado que al realizar el registro de un nuevo usuario, dicho usuario se crea correctamente.
  - Si el usuario nuevo que se va a crear ya existe, no se crea e indica advertencia de que el usuario ya está en uso.
  - Si se deja algún campo vacío en en registro y se intenta registrar, se indica en los campos correspondientes una advertencia de campo vacío.
- Comprobación de login:
  - Se comprueba que al hacer login con usuario correcto, permite entrar en la aplicación sin problemas.
  - Si se intenta loguear con alguno de los dos campos o ambos vacíos, no se permite el login y muestra advertencia de los campos que están vacíos.
- Comprobación de la gestión de las tareas:
  - Se comprueba que si se crea una tarea nueva sin categoría, se permite.
  - Si se crea una tarea nueva con alguna categoría, se crea correctamente.
  - Si se intenta crear una tarea que ya existe, no es posible la duplicidad de la misma.
  - Si se intenta añadir una tarea con el campo de nombre vacío, con categorías con check o sin el, no se permite. Sólo se crean siempre y cuando se introduzca su nombre.
  - Si se modifica una tarea con otro nombre y categorías distintos a las iniciales, se permite el cambio correctamente.
  - Si se modifica una tarea con el mismo nombre o sin introducir nombre, no se permite la modificación, independientemente del cambio que se realice en las categorías.

Do you know SOLID?

Realicé mediante la plataforma de Codely.tv un curso acerca de estos 5 principios, del cual dispongo el título de la plataforma.

Se basa como comento anteriormente en varios principios:

1. S: Principio de responsabilidad única, es decir, que un objeto debería tener responsabilidad única.
2. O: Principio abierto/cerrado: entidades del software abiertas para extensión pero cerradas a modificaciones.
3. L: Sustitución de Liskov: reemplazar objetos del programa por instancias de sus substipos sin alterar el funcionamiento de la aplicación.
4. I: Segregación de interfaz: muchas interfaces cliente específicas con mejores que una de propósito genera.
5. D: inversión a la independencia: Se debe depender de abstracciones y no de implementaciones.

How would you improve your solution? Mention next steps to be considered.

Con más tiempo para continuar con el desarrollo de mi aplicación, se podría mejorar la inclusión de tareas, de modo que en lugar de un formulario simple de nombre y categorías, se pudiera implementar tareas con mucho más contenido, como podrían ser incrustación de fotos, campos de edición de texto, videos embebidos…

También se podría realizar una arquitectura de entidades diferentes para poder incluir en las tareas otros usuarios para compartirlos con ellos y que dichos usuarios pudieran ser participantes de tareas que no sean de su propiedad.

Por último, otra de las funciones que incluiría, sería uno o varios usuarios administradores/moderadores, para que pudieran gestionar la actividad de los usuarios normales y evitar problemas por ejemplo en el caso de que se realizase la inclusión de la arquitectura nueva de entidades comentada anteriormente.

Descarga de la aplicación

Para poder obtener la aplicación hay dos enlaces a los que podemos acceder:

- Github: https://github.com/Carlos2492/gestorTareasConLogin.git.
- Drive: https://drive.google.com/file/d/1q7i4flty420FvJhXRLn7FPNJxRNPAsKL/view?usp=sharing.