BLOG PERIODISTICO

PRACTICA / EXAMEN

PHP

BLOG PERIODISTICO

Este proyecto consiste en crear un blog periodístico donde cualquier usuario se puede registrar para crear noticias, pero con un determinado filtro, donde solo el editor es el que dedique que contenidos se publican y cuales no.

LISTA DE REQUISITOS:

Aunque en el mercado existen numerosos sistemas de administración de contenidos, en algunas ocasiones la creación de un sistema propio ofrece un conjunto de ventajas que hacen que sea la opción más aconsejable. Un sistema propio ofrece más flexibilidad y puede desarrollar y cubrir con más exactitud las necesidades y requisitos del sistema que desea implementar. Así pues, en esta ocasión la elección escogida es el desarrollo propio de un sistema de administración de contenidos en lugar de adquirir o implementar un sistema de los existentes en el mercado.

WorldNews es un medio de comunicación online y cuenta con varias secciones de noticias: deportes, internacional, local, información meteorológica, etc. La página principal muestra los últimos titulares de cada una de las secciones del medio.

El sistema de administración de contenidos ofrece la posibilidad de que los periodistas puedan introducir y/o modificar sus artículos/reportajes y a los editores la revisión y publicación, además también existirá el rol de administrado, el cual podrá acceder al sistema con todos los roles y privilegios.

El hecho de que un periodista haya introducido una nueva noticia no implica que esta se publique inmediatamente ya que es la figura del editor quien decide si una noticia puede publicase además de poder corregirla y maquetarla.

La visualización de las noticias publicadas será pública y por lo tanto no será necesario autenticarse para poder visualizarlas.

Para la gestión e introducción de los contenidos se establecen tres tipos de usuarios con diferentes roles y tipos de privilegios:

- **Periodista**: Puede introducir y modificar noticias/artículos/reportajes. Solo podrá modificar las noticias que él mismo ha introducido previamente y que todavía no están publicadas. No podrá modificar noticias de otros periodistas.

- **Editor**: Tiene permisos para modificar/corregir/maquetar totas las noticias de todos los periodistas. También decide las noticias que se van a publicar. No puede introducir nuevas noticias.

- **Administrador**: Dispone de permisos para realizar cualquier acción.

Los usuarios Periodista, Editor y Administrador deberán autenticarse para poder interactuar con la aplicación, así pues, deberán acceder al gestor de contenidos a través de una página en la puedan introducir un Login y contraseña.

El usuario de perfil periodista podrá registrarse/darse de alta en el registro de usuarios, así como modificar sus datos. El registro de usuarios contendrá los datos básicos de la persona y formas de contacto, además en el momento de registrarse deberá introducir un nombre de usuario, una contraseña que deberá introducirse en dos ocasiones para verificar que coinciden y por lo tanto se ha introducido correctamente, es decir debe aplicarse un sistema de Introducir contraseña y Repetir contraseña.

Para cada noticia, el periodista deberá introducir los datos siguientes: Título, subtítulo, sección, fecha, texto de la noticia, preferiblemente en una área de texto Rich TextArea (puede incluir html) y opcionalmente la posibilidad de añadir una imagen. Además, deberá introducir un conjunto de palabras clave para así facilitar y agilizar las búsquedas de noticias.

Las noticias introducidas por los periodistas no se verán publicadas en la página principal hasta que el editor indique que la noticia se puede publicar, así pues, el editor podrá modificar/corregir las noticias introducidas por los periodistas e indicar si se puede publicar.

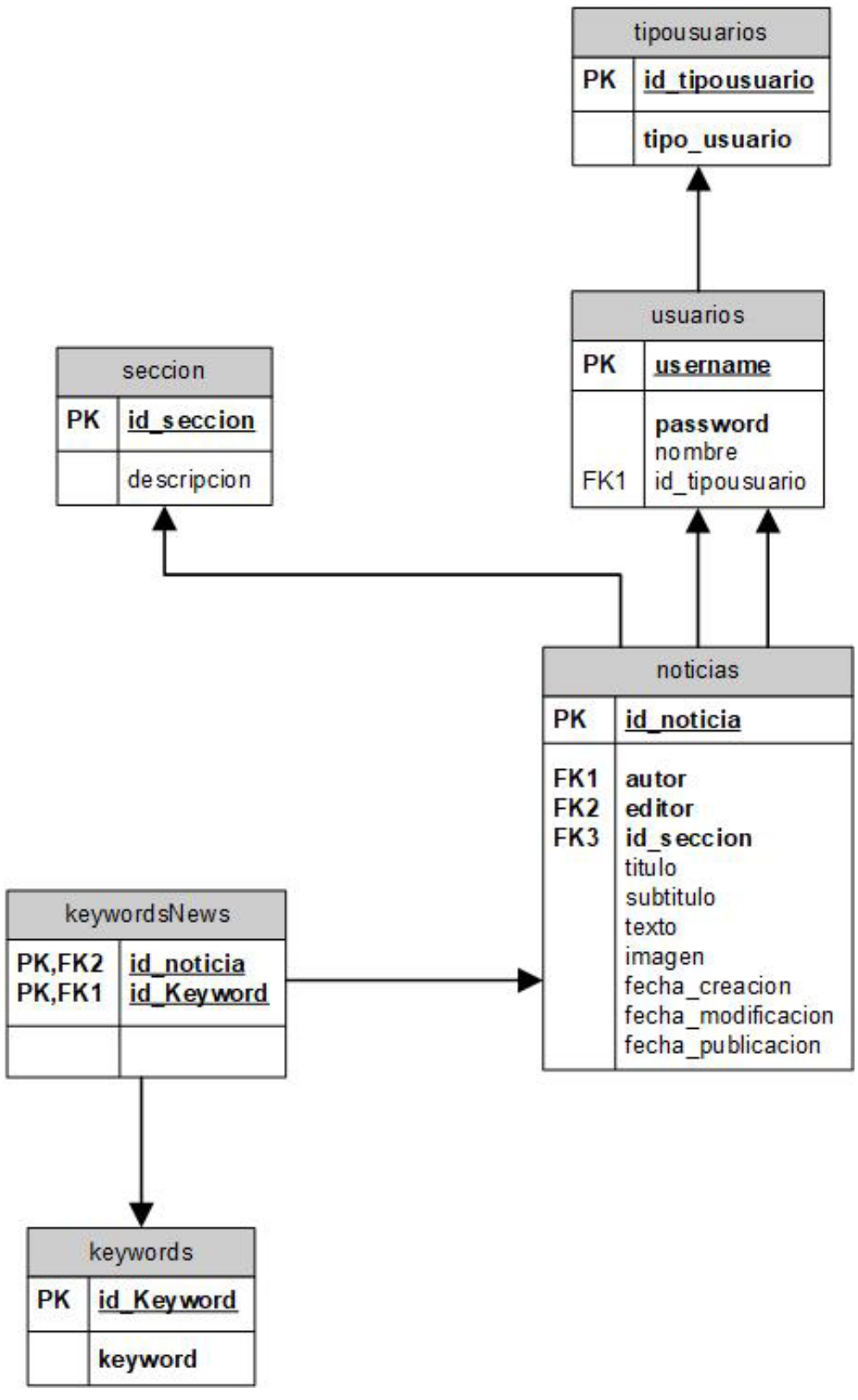
Toda la información de las noticias, usuarios, secciones, palabras clave, etc se almacenará en una base de datos relacional.

El sistema requiere aplicar medidas para mantener la seguridad y la integridad de la información, por ejemplo, en la página de introducción de un reportaje por parte del periodista, debería usarse una transacción para garantizar que se realicen las acciones de inserción de los registros en las tablas de Publicaciones y en la tabla de Palabras Clave de forma que se realicen en bloque. Así pues, la base de datos relacional que debe implementarse debe contener las extensiones necesarias para poder implementar las instrucciones necesarias para realizar transacciones.

Para optimizar las páginas web, debe ejecutarse código en el lado del cliente web en los casos que sea posible. Por ejemplo, en la página de Alta/Registro de Usuarios, se aconseja usar Javascript para verificar que los capos Contraseña y Repetir contraseña tienen el mismo valor.

Para garantizar la accesibilidad de la aplicación, el sitio web debe desarrollarse con interfaces web accesibles, analizando las pautas establecidas y aplicando técnicas de verificación.

ESTRUCTURA DE LA BASE DE DATOS



Requisitos de cada una de las paginas

Inicio / login :

Que haya una pagina iniciar sesión con los campos de nombre de usuario y contraseña, un botón para loguearse y otro botón para registrarse por si el usuario no tiene cuenta.

Registro:

La pagina de registro debe de tener los campos nombre de usuario, el cual debe ser unico, una contraseña, nombre, apellidos, correo electrónico, lugar de residencia y un selector para que el usuario escoja entre hombre y mujer.

Ademas tendrá un botón para poder validar el registro y otro botón para poder volver atrás por si el usuario le hubiera dado al botón de registro sin querer.

- **Blog**

  - **index.php**
    Pagina de inicio que sirve de puente para ir de una pagina a otra por los controladores

  - **Controller (carpeta)**
    Carpeta que contiene cada uno de los controllers relacionados con las vistas

    - **Cart**
      Se encarga de gestionar el carrito de la compra cuando le daños a añadir un producto

    - **news.controller.php**

      - **Index()**
        Te lleva a la pagina de las noticias del usuario logueado

      - **Crud()**
        Te lleva la vista del editor de texto de para generar la noticia

      - **Eliminar()**
        Elimina una noticia ya creada en la base de datos

      - **PaginaNoticia()**
        Te devuelve los datos de una noticia solicitada y los convierte en Json para tratarlos en Javascript para que se vea a través de un modal

      - **Guardar()**
        Cuando se añade o se actualiza una noticia desde el controlador te lleva al Business Object que realizara la operación

      - **keywodsNoticia()**
        Carga las keywords de la noticia seleccionada en el editor de noticias

      - **borrarImagen()**
        Borra una imagen de la galería de imágenes del usuario y del directorio del mismo, ademas de borrar la imagen de la noticia donde haya aparecido.

    - **common.controller.php**

      - **Index()**
        Te lleva a la pagina de inicio donde se muestran todas las noticias publicadas

- **loginregister.controller.php**

  - **Index()**
    te lleva a la pagina de registro de la web

  - **getReristerUser()**
    Redirige a una función el BO que se encarga de recoger todos los datos del formulario pasados por post y los introduce en un objetos de tipo usuario donde después lo envía al DAO que lo inserta en la base de datos

  - **Login()**
    Te lleva a la pagina para poder loguearte.

  - **createDir()**
    si el registro se ha creado con éxito, se crea un directorio con el nombre de usuario para poder guardar las imágenes de las noticias.

  - **checkUser()**
    Recoge los datos de usuario y contraseña pasados por post y los envía al modelos para comprobar que esos valores sean correctos o no con getUserPass
    Si los datos son correctos te le llevara a la tienda, sino de nuevo a la pagina de login.

  - **CerrarSesion()**
    Destruye la variable de sesión y te lleva a la pagina de inicio

- **Model (carpeta)**

En modelo

- **BO (Business Object)**
  En cada Business Object se incluye su respectivo DAO, ademas de iniciar sesión para poder utilizar las variables de sesión

  - **UsersBO**

    - **register()**
      Se obtiene los parámetros del formulario y los setea en un objeto, para ello llama a la function del DAO

    - **checkUsers()**
      Se comprueba que el usuario y la contraseña estén en la base de datos, si coincide se rellenan todos los datos del objetos con los valores del usuarios.

    - **SetterUserDates()**
      Introduce todos los valores del usuario una vez se ha logueado correctamente

  - **NoticiasBO**

    - **TodasNoticiasPublicadas()**
      Muestra todas las publicaciones ya publicados de todos los usuarios

    - **TodasNoticias()**
      Muestra todas las publicaciones de todos los usuarios

- **misNoticias()**
  Muestras las publicaciones del mismo usuario.

- **obtenerId()**
  Se obtiene la id de la noticia y la devuelve para poder hacer uso de una noticia.

- **ElemininarId()**
  Se elimina una noticia en base en base a su id.

- **GaleriaUsuario()**
  Muestra la galería de imágenes de que un usuario en concreto a subido en sus publicaciones

- **EliminarFotoGaleria()**
  Elimina una imagen de su galería y ademas la elimina de la publicación.

- **guardarNoticia()**
  Guarda o actualiza la noticia, se tiene en cuenta si los campos están o no vacíos para setear el objeto noticia para después enviarlo para realizar la transacción en el DAO donde se realizaran todas las comprobaciones para insertar o actualizar la noticia.

- **imagenNoticia()**
  Inserta y publica la imagen de una noticia, comprueba si la imagen proviene de la galería de imágenes del usuario o bien se ha subido por fichero

- **ListadoSecciones()**
  Muestra el listado de las secciones que están disponibles para una noticia

- **mostrarKeywords()**
  Carga todas las keywords disponibles para una noticia

- **SeleccionarKeywordsNoticias()**
  Carga las keywords de una noticia en concreto para poder insertarlas en el formulario de la publicación.

- **DAO (Data Access Object)**
  Todos los archivos DAO contienen un include de los archivos de conexion, su correspondiente Transfer Object y el DataSource compartido

  - **insertNoticia($noticia)**
    inserta en la base de datos la noticia pasado como parámetro el objeto del mismo

  - **insertKewywordsNoticia($keyword)**
    Inserta una keyword en la base de datos que le llega por parámetro

  - **insertKeywordsNews($idkeyword, $idnoticia)**
    Inserta en la base de datos la id de la noticia y su keywords correespondientes en la tabla de keywordsNews

  - **cargarKeywordsNoticia($idnoticia)**
    Muestra todas las keywords de una noticia

- **Mostrar()**
  Muestra todas las noticias creadas por todos los usuarios

- **MostrarNoticias()**
  Muestra por pantalla solo aquella noticias que ya están publicadas por el editor

- **cargarKeywords()**
  Recoge todas las keywords que hay en la base de datos

- **cargarKeywordsSoloKeywords(idnoticia)**
  Recoge todas las keywords que hay en la base de datos vinculadas a una noticia

- **cargarGalerialmagenesAutor($autor)**
  Muestra todas las imágenes que un usuario ha subido en las noticias

- **MostrarMisNoticias($autor)**
  Recoge todas noticias de la base de datos de un usuario en concreto

- **ObtenerId($id)**
  Devuelve como objeto una noticia de la base de datos pasando por parámetro la id del mismo

- **MaxIdNoticia()**
  Obtiene la id de la ultima noticia publicada

- **listaSecciones()**
  Muestra la lista de secciones en las que se pueden publicar noticias

- **Eliminar($idnoticia)**
  Eliminar de la base de datos un producto pasando por parametro la id del producto

- **actualizar($noticia)**
  Actualizar noticia existente

- **actualizarFotoNoticia($noticia)**
  Actualiza foto de la noticia

- **transactionUpdate($newKewwords, $arrayKeywords, $idnoticia, $noticia)**
  Cuando se publica una noticia se tiene que validar que también se publiquen sus palabras clave y para ellos a través de una transacción con beginTransaction, el commit y el rollBack se garantiza que cuando se publique una noticia también se haga todo lo demás.

- **TO (Transform Object)**

- **Conexión**
  contiene toda la información para conectar con la base de datos del servidor.

- **dataSource.php**
  Contiene los métodos para ejecutar las consultar creadas en los DAO

  - **ejecutarConsulta()**
    Ejecuta las consulta del tipo Select.

- **ejecutarActualizacion()**
  Ejecuta las consulta del tipo Insert, Update y delete.

- **insertProduct($product)**
  Inserta en la base de datos un producto pasado como parámetro el objeto del mismo

- **Mostrar()**
  Recoge todos los productos de la base de datos y los pasa a un objeto para que el controlador los maneje

- **Obtener($id)**
  Devuelve como objeto un producto de la base de datos pasando por parámetro la id del mismo

- **Eliminar($id)**
  Eliminar de la base de datos un producto pasando por parámetro la id del producto

- **Actualizar($product)**
  actualiza en la base de datos un producto en concreto

- **conexión.php**
  contiene toda la información para conectar con la base de datos del servidor.

  - **connectdb**()
    Establece la conexión con la base de datos con el sistema de PDO, en la que se debe de poner el servidor, nombre de usuario, contraseña y el nombre de la base de datos con la que se quiera conectar.

- **users.php**

  - **getUserPass()**
    comprueba el usuario y la contraseña y los valida con la base de datos, si los datos son correctos valida la variable de sesión y redirecciona al usuario al inicio de la tienda.

  - **setterUserDates()**
    Inserta los valores pasados por parámetro con la base de datos cuando un usuario se loguea y todo ha sido correcto.

  - **insertRegisterUser()**
    Cuando un usuario se registra desde el controlador envía los datos al modelo y desde la misma función e inserta los datos en la base de datos.

- **Vista (carpeta)**

  - **Home.php**
    Pagina principal donde se muestran todas las noticias publicadas

  - **formNoticia.php**
    Formulario para editar y publicar noticias

  - **Editortexto.php**
    modulo incluido del formulario donde se incluye todos los apartados

  - **misnoticias.php**
    muestras en formato de listado todas las noticias de un usuario.
    las cuales puede borrar, editar o crear según el tipo de usuario.

  - **post.php**
    pagina de la publicación de una noticia

  - **api.php**
    pagina que muestra un json publico para que cualquier usuario pueda acceder a las noticias del canal y distribuirlas libremente.

  - **Css (carpeta)**

    - Reset.css
      Formato el estilo que los navegadores tienen por defecto poniéndolo todo a cero.

    - Style.css
      Se da el estilo que tendrá la misma pagina.

    - Ei.css
      contiene las clase de cada uno de los iconos para poder utilizar con las fuentes dentro de la pagina.

  - **Js (carpeta)**

    - **javascript.js**
      Contiene todas las funciones y lógica de las paginas en común.

    - **editor.js**
      Toda la lógica relacionada con el editor de noticias.

    - **Login.js**
      Contiene las funciones de comprobación para el login y para el registro

  - **Image (carpeta)**

  - **Common (carpeta)**

    - **Header.php**
      Contenido de la cabecera

    - **Footer**
      Pie de la pagina

- **Navbar**
  navegador común de todas las paginas

- **Modules (carpeta)**

  - **Editor**

  - **galeryuser**
    contiene el bloque de la galería de imágenes que el propio usuario ha publicado en sus noticias.

  - **listadonoticiaseditor**
    bloque que contiene la estructura del listado de cada una de las noticias.

- **Fonts(carpeta)**
  Contiene las fuentes que no vienen por defecto en los navegadores.