SALAMANCA360

PARTE LÓGICA

AUTOR: ABDESSAMAD OUBRAHIM AKKOUH
TUTOR: RAFAEL
NOMBRE DEL CENTRO: COLEGIO SANTÍSIMA TRINIDAD

Contenido

En esta memoria del TFG presento nuestra página web llamada Salamanca360. Nuestra página está diseñada para aquellas personas interesadas en conocer nuestra ciudad que es Salamanca. Muchos la visitan por primera vez o están en proceso de visitarla, es decir, que aún están planificando su viaje a nuestra ciudad. Para ello obviamente necesitarán información sobre los principales lugares a los que visitar, actividades a las que acudir, recomendaciones de restaurantes, bares y cafeterías. Y no solo esto, sino que, también necesitarán opciones para alojarse, mapas, rutas, etc. Otro perfil sería el de los estudiantes, que necesitarán también información sobre universidades, bibliotecas, etc. En fin, para todos ellos y más, encontrarán todo aquello que buscan en nuestra página web.

Importante: Esta página web la hemos realizado dos personas. Yo me he encargado del backend y me compañera de la parte del frontend. Tendremos partes en común ya que el proyecto está hecho por los dos.

1. Estudio del problema y análisis del sistema

1.1Introducción

Nuestro proyecto consiste en una página web diseñada para aquellas personas que vienen por primera vez o quieren conocer nuestra ciudad Salamanca. El objetivo de esta página es ayudar a conocer mejor nuestra ciudad con un diseño fácil y vistoso para que el usuario pueda navegar sin dificultad.

Está formado por seis vistas principales: índex, actividades, lugares de interés, gastronomía, ocio y contacto.

- Índex: muestra tres elementos de cada una de las demás vistas, por ejemplo, tres actividades, tres lugares de ocio, etc. Es como una vista general de la página web.
- Actividades: esta vista muestra las diferentes actividades que se pueden hacer en la ciudad.
- Lugares de interés: aquí se muestra los lugares más importantes de la ciudad, como, por ejemplo, monumentos, museos, etc.
- Gastronomía: en esta página se recomiendan restaurantes, bares, etc.

- Ocio: en esta vista se muestra las opciones de entretenimiento que hay en Salamanca, como cines, teatros, discotecas, etc.
- Contacto: aquí tanto usuarios como negocios locales pueden ponerse en contacto con nosotros, tanto para informarnos de algo o intentar publicitarse y así llegar a más gente.

1.2 Finalidad

La finalidad del proyecto es dar a conocer los lugares más importantes de salamanca, dependiendo de los intereses, a aquellos usuarios que llegan por primera vez a salamanca. Lo que queremos conseguir con este sistema, teniendo en cuenta lo anterior, que el usuario se sienta cómodo navegando en nuestra web, que consiga la información que necesita lo más rápido posible, y que esa información le sea valiosa y de gran ayuda.

A parte de esto, personalmente me he encargado del código de la página, lo cual, nuestra finalidad también es conseguir que éste sea ordenado y limpio. Esto es muy importante para nosotros para que los futuros programadores no tengan problema en mantener y entender el código de la página. Y de esta forma asegurar el buen crecimiento de ésta.

En resumen, lo que queremos conseguir, es que, tanto usuarios como futuros programadores no tengan problemas con la página.

1.3 Objetivos

Lo que ofrecemos como tal, es información a nuestros visitantes. El objetivo, es que, en nuestra página encuentren toda la información que necesiten para que puedan conocer y disfrutar de salamanca. Además de eso, queremos ofrecer un espacio en el que todo tipo de negocios puedan publicitarse y así llegar a más gente que esté interesada en sus servicios. Así, ayudamos a que, tanto visitantes como negocios, puedan sacarle el máximo provecho a nuestro proyecto.

Los beneficios que pueden tener los negocios con nuestra idea son los siguientes:

- Llegar a más gente: como hemos comentado anteriormente, al permitir que éstos se anuncien en nuestra página, podrán atraer a más clientes y hacerse conocidos en la ciudad.
- Destacar productos y servicios: esto también ayudará a atraer gente interesada en estos productos y servicios.
- Mejorar la reputación del negocio: al aparecer en una página como la nuestra hace que los visitantes confíen y acudan a estos negocios.

Todos estos puntos son importantes para que el negocio tenga un buen desarrollo y crecimiento.

2. Modelado de la solución

2.1 Recursos humanos

En este momento los que nos hemos encargado de la página web somos dos personas.

Abdel (desarrollador backend):

- Responsable de implementar la lógica de la web del lado del servidor y las funciones del sistema.
- Utilizo el framework Laravel para diseñar la estructura backend del sitio web Salamanca360 para garantizar su estabilidad y eficiencia.
- Creación la base de datos necesaria para almacenar la información de la página web.
- Implemento un administrador para modificar, eliminar y añadir información, proporcionando así, una gestión eficiente del contenido del sitio.

Yo (Iris- desarrolladora front-end):

- Como desarrolladora front-end, mi principal responsabilidad es desarrollar la interfaz de usuario del sitio web de Salamanca360, centrándome en la experiencia de usuario y la presentación visual del contenido.
- Cree interfaces de usuario intuitivas y atractivas utilizando herramientas como Visual Studio, HTML, CSS y JavaScript.
- Diseñe y cree diferentes páginas y secciones del sitio web, garantizando una visualización óptima en una variedad de dispositivos y tamaños de pantalla utilizando un enfoque responsivo.
- Implementé formularios interactivos en el front-end con validación de JavaScript para garantizar la integridad de los datos ingresados por el usuario
- Trabaje en colaboración con Abdel para integrar la interfaz de usuario con la funcionalidad de back-end para garantizar una experiencia de usuario uniforme y fluida en todo el sitio.

Para lograr llevar a cobo este proyecto, hemos trabajado en estrecha colaboración para abordar de manera eficaz los problemas del proyecto. Hemos diseñado el prototipo de nuestra página y después, para llevarlo a cabo, hemos ideado una estrategia. Esta estrategia consiste en, primero analizar todos los puntos que necesita nuestra página para cumplir con sus objetivos. Después, avanzar cada uno con su tarea acordada, finalmente volver al primer punto. Y repetimos este ciclo hasta acabar el proyecto.

2.2. Recursos software

PHP.

PHP es un lenguaje de programación para desarrollar aplicaciones y crear sitios web. Está destinado a desarrollar aplicaciones para la web y crear páginas web, favoreciendo la conexión entre los servidores y la interfaz de usuario.

LARAVEL.

Laravel es un framework de PHP que está hecho para ayudarnos a desarrollar aplicaciones en este lenguaje. Esté framework nos ayuda en muchas cosas al desarrollar una aplicación,

por medio de su sistema de paquetes y de ser un framework del tipo MVC (Modelo-Vista-Controlador) esto nos ayuda muchísimo en ciertos aspectos del desarrollo, cómo instanciar clases y métodos para usarlos en muchas partes de nuestra aplicación sin la necesidad de escribirlo y repetirlos muchas veces con lo que eso conlleva a la hora de modificar algo en el código.

XAMPP

Es un servidor web local multiplataforma que permite la creación y prueba de páginas web u otros elementos de programación.

MySQL.

Es un sistema de gestión de bases de datos relacionales, es una de los gestores más usados y conocidos.

Es fácil agregar, actualizar o borrar tablas, relaciones y hacer otros cambios a los datos cuando lo necesites sin cambiar la estructura general de la base de datos ni afectar las aplicaciones existentes.

Visual Studio.

Es un conjunto de herramientas de desarrollo integradas (IDE) desarrolladas por Microsoft que permiten a los programadores crear aplicaciones para distintas plataformas. Este software ofrece diversas herramientas para facilitar el proceso de desarrollo, como depuración de código, integración con sistemas de control de versiones, pruebas automatizadas y asistencia en la escritura de código.

3. Planificación.

3.1. Diseño del proyecto

3.1.1 Código fuente

Rutas

En primer lugar, explicare el código que define varias rutas para diferentes secciones de la página web. Este código se utiliza para enlazar las solicitudes de los usuarios con los métodos correspondientes en el controlador.

“Route::get('index',[\App\Http\Controllers\Controlador::class,'index'])->name('index');

Route::get('actividades',[\App\Http\Controllers\Controlador::class,'actividades'])

Route::get('gastronomia',[\App\Http\Controllers\Controlador::class,'gastronomia'])

Route::get('lugaresInteres',[\App\Http\Controllers\Controlador::class,'lugaresInteres'])

Route::get('ocio',[\App\Http\Controllers\Controlador::class,'ocio'])->name('ocio');

Route::get('contacto',[\App\Http\Controllers\Controlador::class,'contacto'])”

Estas rutas definen una ruta GET llamando al método indicado. Esta sección se encarga de llamar a los métodos en el Controlador cuyo objetivo es mostrar las vistas que va a ver el usuario.

“Route::get('adminActividades',[\App\Http\Controllers\Controlador::class,'adminActividades'])

Route::get('adminGastronomia',[\App\Http\Controllers\Controlador::class,'adminGastronomia'])

Route::get('adminLugaresInteres',[\App\Http\Controllers\Controlador::class,'adminLugaresInteres'])

Route::get('adminOcio',[\App\Http\Controllers\Controlador::class,'adminOcio'])”

Estas rutas, al igual que las anteriores, definen una ruta GET llamando al método indicado. Esta sección se encarga de llamar a los métodos en el controlador cuyo objetivo

es mostrar las vistas que va a ver el administrador de la pagino. A estas solo debe tener acceso el administrador ya que tienen opciones de eliminar, añadir y modificar.

“Route::get('formulario',[\App\Http\Controllers\Controlador::class,'formulario'])->name('formulario');

Route::get('formActividades',[\App\Http\Controllers\Controlador::class,'formActividades'])->name('formActividades');

Route::get('formGastronomia',[\App\Http\Controllers\Controlador::class,'formGastronomia'])->name('formGastronomia');

Route::get('formLugaresinteres',[\App\Http\Controllers\Controlador::class,'formLugaresinteres'])->name('formLugaresinteres');

Route::get('formOcio',[\App\Http\Controllers\Controlador::class,'formOcio'])->name('formOcio');”

Estas rutas, al igual que las anteriores, definen una ruta GET llamando al método indicado. Esta sección se encarga de llamar a los métodos en el controlador cuyo objetivo es mostrar las vistas del formulario para poder añadir actividades, gastronomía, lugares de interés y ocio.

“Route::post('guardarActividades',[\App\Http\Controllers\Controlador::class,'guardarActividades'])->name('guardarActividades');

Route::post('guardarGastronomia',[\App\Http\Controllers\Controlador::class,'guardarGastronomia'])->name('guardarGastronomia');

Route::post('guardarLugaresInteres',[\App\Http\Controllers\Controlador::class,'guardarLugaresInteres'])->name('guardarLugaresInteres');

Route::post('guardarOcio',[\App\Http\Controllers\Controlador::class,'guardarOcio'])->name('guardarOcio');”

Estas rutas definen una ruta POST llamando al método indicado. En este caso utilizamos el método POST ya que enviamos información desde cliente al servidor para ser guardada. Este método es más seguro para enviar datos, ya que los datos no se adjuntan a la URL como en las solicitudes GET, sino que, se envían en el cuerpo de la solicitud, lo que ayuda a mantener la URL limpia y protege los datos de ser expuestos en los registros del servidor o en el historial del navegador.

El método POST se utiliza en este caso también porque es necesario enviar datos complejos al servidor, incluyendo imágenes, que no pueden ser manejados adecuadamente por el método GET. El método GET está diseñado principalmente para solicitudes de recuperación de datos y tiene limitaciones significativas en cuanto a la cantidad de datos que puede enviar.

Esta sección se encarga de llamar a los métodos en el controlador cuyo objetivo es guardar la información recogida de los formularios en la base de datos.

“Route::delete('destroyActividad/{id}',[\App\Http\Controllers\Controlador::class,'destroyActividad'])->name('destroyActividad');

Route::delete('destroyGastronomia/{id}',[\App\Http\Controllers\Controlador::class,'destroyGastronomia'])->name('destroyGastronomia');

Route::delete('destroyLugaresInteres/{id}',[\App\Http\Controllers\Controlador::class,'destroyLugaresInteres'])->name('destroyLugaresInteres');

Route::delete('destroyOcio/{id}',[\App\Http\Controllers\Controlador::class,'destroyOcio'])->name('destroyOcio');”

Estas rutas definen una ruta DELETE llamando al método indicado. En caso utilizo este método ya que mi objetivo es eliminar una actividad, gastronomía, lugares de interés u ocio con el id indicado. Al llamar a estas rutas se está llamando a los métodos del controlador cuyo objetivo es eliminar información desde la base de datos.

“Route::get('editActividad/{id}',[\App\Http\Controllers\Controlador::class,'editActividad'])->name('editActividad');

Route::get('editGastronomia/{id}',[\App\Http\Controllers\Controlador::class,'editGastronomia'])->name('editGastronomia');

Route::get('editLugaresInteres/{id}',[\App\Http\Controllers\Controlador::class,'editLugaresInteres'])->name('editLugaresInteres');

Route::get('editOcio/{id}',[\App\Http\Controllers\Controlador::class,'editOcio'])->name('editOcio');”

Estas rutas definen una ruta GET llamando al método indicado. Esta sección se encarga de llamar a los métodos en el controlador cuyo objetivo es mostrar las vistas de editar, que son formularios también, para editar actividades, gastronomía, lugares de interés u ocio con el id indicado.

“Route::put('updateActividades/{id}',[\App\Http\Controllers\Controlador::class,'updateActividades'])->name('updateActividades');

Route::put('updateGastronomia/{id}',[\App\Http\Controllers\Controlador::class,'updateGastronomia'])->name('updateGastronomia');

Route::put('updateLugaresInteres/{id}',[\App\Http\Controllers\Controlador::class,'updateLugaresInteres'])->name('updateLugaresInteres');

Route::put('updateOcio/{id}',[\App\Http\Controllers\Controlador::class,'updateOcio'])->name('updateOcio');”

Estas rutas definen una ruta PUT llamando al método indicado. Esta sección se encarga de llamar a los métodos del controlador cuyo objetivo es modificar los datos en la base de datos con el id indicado. El método PUT se utiliza para actualizar información ya existente en el servidor. El método PUT generalmente se espera que proporcione una representación completa del recurso que se está actualizando.

Controlador

En segundo lugar, explicare el código del controlador. Este es una clase que se encarga de manejar solicitudes HTTP de nuestra aplicación. Los controladores son los intermediarios entre los modelos (lógica de negocio y datos) y las vistas (interfaz del usuario).

“namespace App\Http\Controllers;”

Esta es la primera línea de nuestro Controlador, y define el espacio de nombres para esta clase. Lo que viene seguido del namespace indica que esta clase se encuentra en ese directorio.

“use Illuminate\Http\Request;”

Esta línea se utiliza para importar la clase Request para poder utilizarla en nuestro Controlador sin tener que escribir el namespace completo cada vez que la necestemos.

Esta clase en Laravel se utiliza para encapsular toda la información sobre una solicitud HTTP enviada a nuestra aplicación. Esto quiere decir que incluye datos del formulario, cookies, archivos subidos, etc.

“use App\Models\Actividad;

use App\Models\Gastronomia;

use App\Models\LugarInteres;

use App\Models\Ocio;”

Estas líneas Importan los modelos que se encuentran en ese directorio, y representan una tabla en la base de datos. También proporcionan una interfaz para interactuar con dichas tablas. Otro dato importante, es que, los nombres de los modelos suelen estar en singular y en mayúscula inicial, mientras que los nombres de las tablas están en plural.

Cuando importamos estos modelos, estamos indicando que los vamos a utilizar en nuestro controlador. Esto es imprescindible porque Laravel necesita saber a qué modelo nos estamos refiriendo para poder realizar operaciones sobre la base de datos.

”use Illuminate\Support\Facades\File;”

Esta línea importa la fachada Redirect desde el namespace especificado. Al hacerlo, podemos utilizar Redirect en nuestro controlador para realizar redirecciones sin tener que escribir el namespace completo.

La fachada Redirect proporciona muchos métodos que no serán útiles para redirigir a nuestros usuarios a la vista que nos convenga en ese momento.

“use Illuminate\Support\Facades\File;”

Esta línea importa la fachada File desde el namespace indicado. Permite el acceso a varias operaciones de manejo de archivos. En Laravel es una herramienta poderosa para manejar archivos dentro de nuestra página web. Nos Proporciona métodos fáciles de usar para realizar operaciones comunes como leer, escribir, mover y eliminar archivos. Importarla en nuestro controlador nos permite acceder a esta funcionalidad de manera eficiente, lo que nos facilita la manipulación de archivos, la mejora de la organización y mantenibilidad del código.

Vistas Principales

```
function index()
   {
      return View('index');
   }
```

Esta es una funcion del controlador. Su único objetivo es devolver una vista cuyo nombre coincide con el del método y la ruta que ya hemos explicado anteriormente.

View() es una función que se utiliza para cargar y devolver una vista Blade. La carga se hace desde el directorio resources/views de la página. Pasamos el nombre de la vista como argumento. Por ejemplo, en nuestro caso view('index') buscará esta vista que se llama index.blade.php en el directorio antes mencionado.

```
“public function actividades()
   {
      $actividades= Actividad::all();
      return View('actividades.actividades',[
         'actividades'=> $actividades
      ]);
   }”
```

El objetivo de esta función es cargar todas las actividades de la base de datos utilizando el modelo Actividad y luego llama a la vista actividades.

Este código utiliza el modelo mencionado anteriormente, para realizar una consulta a la base de datos, y así, obtener todos los registros de la tabla actividades como ya expliqué anteriormente. Esto se consigue gracias a la función all() que devuelve una colección de estos registros que en este caso se los asigna a la variable $actividades.

En la función view cargamos la vista actividades. Esta vista se encuentra en resources\views\actividades\actividades.blade.php, por esta razón, se le pasa a dicha función "actividades.actividades" ya que la vista actividades.blade.php se encuentra dentro de la carpeta actividades en views.

En este caso, también se le pasa un array con una clave "actividades" que contiene la colección de actividades obtenida en el paso mencionado anteriormente. Esto se hace para pasar datos desde el controlador a la vista, y de esta manera, puede acceder a ella y mostrar la información de las actividades en la interfaz de usuario.

Los tres métodos siguientes (gastronomía, lugaresInteres y ocio) se hace exactamente lo mismo. El objetivo también es cargar toda la información de su tabla correspondiente de la base de datos utilizando su modelo mencionado en el método y luego llamando a la vista que se carga en la función view.

En resumen, los métodos siguen el mismo patrón para cargar y pasar datos a las vistas, solo que se están utilizando diferentes modelos y vistas para presentar diferentes tipos de información en la interfaz de usuario.

Vistas administrador

Los métodos adminActividades, adminGastronomia, adminLugaresInteres y adminOcio hacen lo mismo que los métodos anteriores, solo que, en este caso las vistas cargadas en la funcion View son las que únicamente puede y debe ver el administrador de la página, ya que tienen opciones de editar, eliminar, modificar, etc. El objetivo de estos métodos anteriormente mencionados es el mismo que el de actividades, gastronomía, lugaresInteres y ocio ya que también cargan toda la información de su tabla

correspondiente de la base de datos utilizando su modelo mencionado en el método y luego llamando a la vista que se carga en la función view.

Vistas de un solo elemento

```
“public function mostrarVistaGastronomia($id)
   {
      $VistaGastronomia=Gastronomia::find($id);
      return View('gastronomia.vistaGastronomia',[
         'VistaGastronomia'=> $VistaGastronomia
      ]);
   }”
```

El objetivo de este método es obtener un elemento cuyo id coincida con el que se le proporciona.

Empieza buscando en la tabla del modelo Gastronomía dentro de la base de datos el registro que tenga el id que se busca utilizando el método find, que es el encargado de realizar esta búsqueda. Después de encontrar el registro, Se lo asignamos a una variable llamada VistaGastronomia.

Luego, cargamos nuestra vista llamada “gastronomia.vistaGastronomia” utilizando la función View que ya explicamos. Y le pasamos como argumento también el objeto que se almacena en la variable VistaGastronomia.

Esto hará que la vista “gastronomia.vistaGastronomia” tenga acceso al objeto que se almacena en la variable mencionada, para que así, pueda mostrar la información de dicho objeto de Gastronomía.

Los métodos mostrarActividad, mostrarLugarInteres y mostrarOcio hacen exactamente lo mismo. Su objetivo es obtener un elemento cuyo id coincida con el que se le proporciona y hacer que la vista que se carga en la función View tenga acceso a dicho elemento para poder mostrar su información.

Formularios

```
“public function formulario()
   {
      return View('administrador.formulario');
   }”
```

Este método tiene como objetivo mostrar la vista que se le especifica dentro de nuestra página web, concretamente la vista del formulario del administrador. Cuando se llama esta función, se carga y se muestra la vista que se encuentra en la carpeta de administrador dentro de views en. Esta vista solo la puede ver el administrador, ya que con este formulario se añadirá información a la tabla que desea el administrador en la base de datos, para, posteriormente mostrar esa información al usuario.

Los métodos formActividades, formGastronomia, formLugaresInteres y formOcio también tienen como objetivo mostrar la vista que se les especifica dentro de nuestra página web. Estas vistas se encuentran también en la carpeta administrador dentro de views.

Cada formulario tiene como función recoger e enviar la información que se le indica a las tablas, cuyo nombre coinciden con el de los métodos, dentro de la base de datos.

Métodos Guardar

Ahora explicaré el método llamado guardar Actividades. El objetivo de este método es procesar los dataos que se le envían desde el formulario y guardarlos en la base de datos. Este proceso se hace mediante la creación de un nuevo objeto del modelo de Actividad, y representa la tabla cuyo nombre coincide, pero en plural dentro de la base de datos. Hacer esto equivale a preparar un registro de dicha tabla para posteriormente llenarlo con datos que se correspondan con sus columnas que son los mismos que los del formulario mencionado anteriormente.

Lo siguiente es asignar los valores del formulario a los campos del objeto ya creado. Para lograr esto debemos poder acceder a los datos enviados por el objeto $request ya que contiene todos los datos del formulario. Asignamos los campos nombre, ubicación, ubicación_enlace y enlace del formulario a los campos correspondientes del objeto del modelo Actividad. Así aseguramos que los datos introducidos por el usuario en el formulario se ajusten correctamente en objeto.

Ahora pasamos a manipular el archivo de imagen que el usuario ha subido desde el formulario. Este archivo se obtiene mediante el método file desde el objeto $request usando el nombre del campo del formulario que es imagen. Cuando ya lo conseguimos, guardamos el nombre del archivo en el campo imagen_ruta del objeto $actividad para poder acceder a él en un futuro y mostrar la imagen.

Después de realizar el paso anterior, movemos la imagen a la carpeta imágenes que se encuentra dentro de public. Conseguimos esto gracias al método move, le asignamos como parámetros la ruta donde queremos dejar el archivo con el nombre que lo queremos guardar, que en este caso debe coincidir con el nombre de la imagen del objeto $actividad. De esta forma aseguramos que la imagen se guarda de forma organizada y accesible.

Continuamos asignando los valores de descripcionMin, descripcionMax, teléfono y precio del formulario a los campos correspondientes del objeto $actividad. Igual que los campos anteriormente mencionados, cogemos estos datos desde el objeto $request, permitiendo así, su almacenamiento.

Después de que todos los campos de nuestro objeto $actividad se hayan completado correctamente, llamamos al método save del modelo Actividad. Este método guarda toda la información que le hemos proporcionado en la base de datos.

Y finalmente, después de guardar todos los datos, se redirige a los usuarios de vuelta a la vista principal de formularios. Esto se consigue gracias al método redirect.

Los métodos guardarGastronomía, guardarOcio y guardarLugaresInteres hacen exactamente lo mismo. Toman los datos de sus formularios correspondientes, se asignan a nuevos registros de la base de datos mediante sus modelos y luego también se redirigen a la vista principal de formulario.

Métodos editar

```
public function editActividad($id)
   {
      $actividad = Actividad::find($id);
      return View('administrador.editar.actividades',[
      'actividad'=>$actividad
      ]);
   }
```

El objetivo de este método este método es facilitar la edición de una actividad concreta de nuestra página web. Debe ser capaz de llevarnos a la vista de editar con la información de la actividad cuyo id se le pasa como parámetro.

Después de conseguir la actividad deseada, se la enviamos a la vista para poder visualizar su información.

Métodos Guardar cambios

En esta sección del código vamos a explicar cómo se guardan los cambios en la base de datos después de editar.

Esto es lo mismo que en los métodos guardar, ya que también recogemos la información desde los formularios de las vistas de editar a las que puede acceder el administrador.

También utilizamos el objeto $request para recoger la información que envía el formulario y de esta forma poder asignársela al objeto del modelo indicado.

La diferencia en este caso está en que el objeto del modelo que modificamos no es un objeto creado nuevo, sino que, con el método find recogemos el que se identifique con el id que se le envía como parámetro a los métodos guardar cambios.

Y aquí es cuando modificamos los campos de dicho objeto con su información correspondiente.

Destroy

En esta sección del controlador explico los métodos “destroy” que hemos realizado. El objetivo de estos métodos es eliminar un objeto, que pertenezca a un modelo en concreto y a un registro de la tabla de dicho modelo, cuyo id se identifique con el que le pasamos como parámetro en el método.

Empezamos creando una variable $actividad que le asignamos como valor un objeto del modelo “Actividad”. Este objeto debe coincidir con el id que le pasmos como parámetro a la función, y lo conseguimos gracias al método find de nuestro modelo.

Después en la línea siguiente le decimos que elimine este objeto con el método delete de dicho modelo. Gracias a este método podemos eliminar el registro cuyo id sea el del parámetro de la función mencionado anteriormente.

Finalmente terminamos redirigiendo al usuario a la vista desea gracias al método redirect. Este método se encarga de redirigir al usuario a dicha vista después de realizar todos los pasos anteriores.

Estos mismos pasos se repiten con todos los métodos de “destroy”. La única diferencia está en que cada método debe eliminar registros de la tabla de la base de datos del modelo que se le asigna.

Vistas

En esta sección no me voy a entretener mucho ya que en muchas vistas el objetivo es el mismo. Su función en este caso es mostrar la información que hay en nuestra base de datos.

Voy a explicar principalmente los foreach, que son bucles que van a recorren el array, que, en este caso, es el que le enviamos desde el controlador, como, por ejemplo, $actividades. Estos array, como ya he explicado en la sección del controlador antes, contienen objetos del modelo que se les asigna.

La información que va a mostrar este bucle en cada tarjeta es una imagen del objeto en concreto, el nombre, una descripción pequeña y un botón “ver más” que al clicarlo lleva a una vista con información más extensa.

Esto se repetirá en la mayoría de las vistas. Por esto mismo, no me entretendré explicando esta sección de las vistas.

4. Fase de pruebas

4.1 Pruebas realizadas

En este caso hacer las pruebas no he utilizado ningún programa, sino que, cada vez que realizaba un método lo probaba y en caso de dar algún error lo identificaba y después lo corregía. Esto me ha permitido mejorar la ejecución de nuestra página web. A continuación, voy a explicar esto con más detalle.

1. Primero escribía el código para cada función de nuestra pagina web.
2. Después de escribir el código, lo ejecuto y pruebo si funciona. En muchos casos salta con más errores de sintaxis que de lógica.
3. Identifico dichos errores.
4. Corrijo los errores identificados en el paso anterior.

Esto se repite hasta que el código sea el deseado y la información que muestra sea correcta.

Ejemplos de pruebas

Creación de Actividades.

Se creaba una nueva actividad y se verificaba que todos los campos se guardaban correctamente en la base de datos.

Después se probaba si estos datos se mostraban correctamente en las vistas.

Si no es así se, identifico el problema.

Y por último se corrige.

4.2 Posibles mejoras del código

Hay varias mejoras en el código para optimizar la estructura, organización y mantenibilidad de éste. Aquí explico algunas de ellas:

1. Tener varios controladores en vez de tener solo uno: Ahora mismo el código solo utiliza un controlador para todos los modelos y también para el administrador, lo cual, hace que el código sea desordenado y difícil de mantener.
   Para arreglar esto, lo mejor es dividir nuestro controlador principal en varios controladores, cada uno se encargaría de cada modelo, otro del administrador y otro para la parte del usuario.
   Esto beneficiaría la organización del código, facilitaría la corrección de errores y hace que se pueda mantener mejor dicho código.

2. Tener varios resources en la parte de rutas: Esto simplificaría la definición y gestión de las rutas, haría más fácil la comprensión de la estructura de rutas y reduce mucho las líneas de código en esta parte.

Esto es importante ya que es importante asegurarse de que el código esté bien organizado y estructurado para mejorar la mantenibilidad y escalabilidad de la página web. Esto no solo incluye la organización de controladores y rutas, sino también la correcta utilización de modelos, vistas y otros componentes de Laravel.

5. Conclusiones finales

5.1. Grado de cumplimiento de los objetivos fijados.

Nuestro proyecto cumple con todos los objetivos que en un principio hemos establecido ya que podemos añadir, modificar, eliminar y ver todos los elementos de la base de datos.

Se ha logrado un desarrollo funcional de nuestra página web ya que aparte de conseguir todas las funcionalidades mencionadas anteriormente, también hemos conseguido que, en cuanto a lo visual sea bonito.

El objetivo era implementar todas las funcionalidades requeridas de manera completa y funcional. Y no solo esto, sino que también proporcionar a los usuarios una página web de su agrado.

5.2. Propuesta de modificaciones o ampliaciones futuras del sistema implementado

Para mejorar y ampliar nuestra página web se podemos añadir varias modificaciones que mejorarían la experiencia de usuario y la funcionalidad de ésta. Aquí explico posibles mejoras:

1. Un filtrar: Esto ayudaría a nuestros visitantes a encontrar lugares que encajen con sus gustos y exigencias mucho más rápido que sin dicho filtro. Éste podría ser una buena herramienta ya que también podemos hacer que filtre por los lugares más cercanos al usuario, lo cual, lo haría mucho más funcional y mejoraría la experiencia de nuestros usuarios.
2. Que los usuarios se puedan registrar y guardar sus sitios favoritos, esto permite que los usuarios puedan crear cuentas y así conseguir que puedan guardar sitios de ocio, gastronomía, ciudades y ocio nocturno para posteriormente acceder a ellos rápidamente ahorrándoles tiempo.
3. Gracias al punto anterior se podría desarrollar un sistema que recomiende al usuario lugares y actividades basándose en los sitios que le gusten. Esto se podría hacer creando algoritmos de recomendación para sugerir dichos lugares y actividades.
4. Desarrollar una aplicación móvil con las mismas utilidades. Esto ayudaría a llegar a más gente ya que muchos prefieren acceder a aplicaciones móviles en vez de páginas web porque lo consideran más cómodo y accesible.

Estas modificaciones futuras ayudarán a mejorar la funcionalidad de nuestra página web, ya que los usuarios la verán como una página más completa.

6. Documentación del sistema desarrollado

6.1. Manual de uso.

En nuestro proyecto Salamanca360 hemos creado un manual de uso para ayudar al usuario a navegar y utilizar las distintas secciones de nuestra plataforma web.

Nuestra plataforma web está diseñada para que sea una herramienta útil y practica y que todo el mundo de cualquier edad pueda navegar sin problemas en nuestra página web.

Ofrecemos una variedad de secciones dedicadas a diferentes aspectos de la ciudad como actividades, lugares de interés, gastronomía, ocio nocturno y contacto que ya mencionamos antes. A continuación, explicaremos el manual.

Página de Inicio

La primera página que ve el usuario es esta lo primero que van a ver es el logo con el menú de navegación que está formado por actividades, lugares de interés, gastronomía, ocio nocturno y contacto si hace clic en alguna de ellas lo llevara a la página seleccionada.

Debajo de menú el usuario encontrará un carrusel donde podrá interactuar para ver una pequeña descripción tanto escrita como visual de lo que ofrecemos en la página web (las vistas del menú excepto el contacto).

Después de esto el usuario encontrara tres tarjetas, que son, las más destacadas de cada vista. Cada tarjeta se compone de una imagen, el nombre de la actividad, lugar, bar o discoteca. A continuación, una pequeña descripción de dicho lugar y un botón llamado" ver-más" donde el usuario pueda clicar y acceder a una vista para obtener más información sobre dicho lugar.

Debajo de estas tres tarjetas podrá encontrar un botón donde podrá acceder a la sección de dichas tarjetas.

Esto se repite el mismo número de veces que secciones tenemos en nuestra plataforma web hasta llegar al pie de página. En éste encontrara un logo y un certificado donde le indicamos y aseguramos que sus derechos se trataran según la ley de protección de datos.

Página de Actividades

En esta sección se encontrará también el logo, que clicando en el podrá regresar a la página de inicio. Con esto, se encontrará también el menú de navegación que está formado por actividades, lugares de interés, gastronomía, ocio nocturno y contacto si hace clic en alguna de ellas lo llevara a la página seleccionada.

A continuación, se ve una imagen llamativa y descriptiva de la sección en la que se encuentra, que es, en este caso, actividades.

Haciendo Scroll se encontrará las tarjetas sólo relacionadas con dicha sección que como ya explicamos antes cada tarjeta se compone de una imagen, el nombre de la actividad y a continuación, una pequeña descripción de dicho lugar y un botón llamado" ver-más" donde el usuario pueda clicar y acceder a una vista para obtener más información sobre el lugar.

También encontraras el pie de página que es el mismo que la página principal.

Página de Lugares de Interés

Esto es exactamente igual que la página de actividades, pero con la información específica de lugares que ver en Salamanca.

Página de Gastronomía

Esto es exactamente igual que la paginas anteriores, pero con la información específica de gastronomía.

Página de Ocio nocturno

Esto es exactamente igual que la paginas anteriores, pero con la información específica de del ocio de Salamanca.

Página de Contacto

Esta página es muy importante para nuestro proyecto ya que es la parte en la que los usuarios podrán ponerse en contacto con nosotros para mostrarnos sus dudad, sugerencias, comentarios, etc.

En esta sección se encontrará también el logo, que clicando en el podrá regresar a la página de inicio. Con esto, se encontrará también el menú de navegación que está formado por actividades, lugares de interés, gastronomía, ocio nocturno y contacto si hace clic en alguna de ellas lo llevara a la página seleccionada como ya explicamos anteriormente.

A continuación, se ve una un formulario que está formado por los siguientes campos:

- Nombre: donde el usuario nos indicará su nombre para que se en caso de respuesta su petición podamos dirigirnos a él correctamente generando así cercanía.
- Apellidos: Aquí nos indicará sus apellidos o apellido por la misma razón que la del punto anterior.
- Email: En este campo nos proporcionará su email para que en caso de aceptar la petición contactar con él por éste mismo.
- Confirmación del Correo electrónico: este campo es crucial para asegurarnos de que el email anteriormente mencionado sea correcto y el usuario no se equivoca. El objetivo es repetir el correo electrónico, en caso de ser idéntico al anterior quiere decir que hay bastantes probabilidades de que el usuario nos indica un correo correcto, y en caso contrario, querrá decir que éste es erróneo y al enviar el formulario saltará un error para que pueda corregirlo.

- Descripción: en esta sección del formulario se nos proporcionará la información que el usuario nos quiera dar.

También encontraras el pie de página que es el mismo que la página principal.

7. Bibliografía

https://www.youtube.com/watch?v=8dRi0qWEx0k

https://rockcontent.com/es/blog/php/

https://openwebinars.net/blog/que-es-laravel-caracteristicas-y-ventajas/

https://www.dongee.com/tutoriales/que-es-xampp/#:~:text=Como%20te%20mencionamos%20antes%2C%20Xampp,facilitan%20la%20experiencia%20al%20desarrollador.

https://datascientest.com/es/mysql-el-sistema-de-gestion

https://cursos.frogamesformacion.com/pages/blog/que-es-visual-studio

https://asana.com/es/resources/project-design

https://publuu.com/es/knowledge-base/como-escribir-un-manual-de-usuario/

https://riunet.upv.es/bitstream/handle/10251/10273/Memoria.pdf