UNIVERSIDAD GERARDO BARRIOS

FACULTAD

CIENCIA Y TECNOLOGÍA

ASIGNATURA

PROGRAMACIÓN COMPUTACIONAL IV

CARRERA

INGENIERÍA EN SISTEMAS Y REDES INFORMÁTICAS

DOCENTE

ING. WILLIAN MONTES

ESTUDIANTES:

JEFFERSON JOSUÉ ESPERANZA ORTIZ- SMSS132422

EDWIN ALEXANDER VILLALTA ORTIZ - SMSS022022

GERSON NAHUM ARGUETA HERNANDES- SMSS021722

ÁNGEL JOSUÉ GUEVARA PORTILLO - SMSS015622

SOFÍA MARGARITA ROMERO RODRIGUEZ- SMSS042622

CÉSAR ALEXANDER ROMERO VAZQUES- SMSS078821

SAN MIGUEL, EL SALVADOR

Contenido

INTRODUCCIÓN

En el dinámico mundo de los servicios profesionales, la plataforma Talento emerge como un innovador mercado digital que conecta a proveedores especializados con clientes que buscan contratar servicios de alta calidad. Talento ofrece una interfaz intuitiva y segura para realizar transacciones, facilitando una experiencia de compra y venta fluida y confiable. A través de Talento, los usuarios pueden explorar diversas categorías de servicios, acceder a perfiles detallados de proveedores, comunicarse directamente y acordar términos de manera eficiente. Esta plataforma no solo optimiza la búsqueda y contratación de servicios profesionales, sino que también establece un entorno colaborativo y transparente que beneficia tanto a proveedores como a clientes.

OBJETIVO GENERAL

Definir una plataforma digital que, alineada con nuestra misión, simplifique el proceso de contratación de servicios profesionales, facilitando la conexión eficiente entre proveedores altamente especializados y clientes a nivel mundial, promoviendo la colaboración y el crecimiento profesional.

OBJETIVOS ESPECÍFICOS

I. Establecer un sistema que permita a los usuarios explorar diversas categorías de servicios, acceder a perfiles detallados de proveedores y comunicarse directamente con ellos para acordar términos y condiciones.
II. Establecer rigurosos procesos de selección y evaluación para garantizar la calidad de los proveedores y los servicios ofrecidos en la plataforma.
III. Implementar medidas de seguridad avanzadas para proteger los datos personales y financieros de los usuarios, así como fomentar la comunicación transparente y las evaluaciones verificadas.
IV. Continuar diversificando las categorías de servicios ofrecidos en la plataforma para satisfacer las necesidades de diferentes sectores y campos profesionales.

ALCANCE

El alcance del trabajo abarca el desarrollo completo de la plataforma Talento, un mercado digital innovador diseñado para la compra y venta de servicios profesionales. Talento se concibe como una solución integral que proporciona una experiencia de usuario intuitiva y segura, garantizando una conexión fluida entre proveedores altamente especializados y clientes de todo el mundo. Esta plataforma global ofrecerá una amplia variedad de servicios, desde consultoría

empresarial hasta diseño gráfico y desarrollo web, cubriendo las necesidades laborales de una amplia gama de industrias.

TECNOLOGÍAS

En el desarrollo de la plataforma "Talento", se han utilizado diversas tecnologías que han sido seleccionadas meticulosamente para asegurar un rendimiento óptimo, una escalabilidad adecuada y una experiencia de usuario fluida.

- PHP (Versión 8.2.12)

Descripción y Motivo de Elección:

PHP es un lenguaje de programación de código abierto ampliamente utilizado para el desarrollo web. La versión 8.2.12 de PHP ha sido elegida debido a sus mejoras significativas en rendimiento y seguridad respecto a versiones anteriores. Esta versión incluye características avanzadas como la mejora en el manejo de errores y excepciones, la optimización de la memoria y la inclusión de nuevas funcionalidades que facilitan el desarrollo ágil y eficiente de aplicaciones web.

- Uso en el Proyecto:

En "Talento", PHP se utiliza como el lenguaje principal del lado del servidor. Su rol es crucial para manejar la lógica de negocio, procesar las solicitudes de los usuarios y comunicarse con la base de datos. PHP se encarga de generar el contenido dinámico de la aplicación y de garantizar una interacción fluida entre el frontend y el backend.

- Laravel (Versión 11.8.0)

Descripción y Motivo de Elección:

Laravel es un framework de PHP conocido por su elegancia y simplicidad, lo que facilita el desarrollo rápido y estructurado de aplicaciones web. La versión 11.8.0 de Laravel ha sido seleccionada por sus características avanzadas, como su potente motor de plantillas, su ORM (Eloquent) que simplifica las operaciones de la base de datos, y su robusto sistema de autenticación y autorización.

- Uso en el Proyecto:

Laravel se utiliza en "Talento" para estructurar y organizar el código del backend de manera eficiente. Proporciona una base sólida para la construcción de la lógica de negocio y la gestión de rutas, controladores y vistas. Además, Laravel facilita la integración de servicios externos y el manejo de tareas comunes como el envío de correos electrónicos, la gestión de sesiones y la validación de datos.

- PostgreSQL (Release 16.2)

Descripción y Motivo de Elección:

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, conocido por su robustez, escalabilidad y conformidad con el estándar SQL. La versión 16.2 de PostgreSQL ha sido elegida debido a sus mejoras en rendimiento, soporte para JSON y sus capacidades avanzadas de replicación y recuperación de datos.

- Uso en el Proyecto:

En "Talento", PostgreSQL actúa como la base de datos principal, almacenando toda la información relacionada con los usuarios, proveedores, servicios y transacciones. Su capacidad para manejar grandes volúmenes de datos de manera eficiente y su flexibilidad para soportar consultas complejas son fundamentales para asegurar que la plataforma funcione de manera rápida y confiable. Además, PostgreSQL permite mantener la integridad y seguridad de los datos, lo cual es crucial para ganar la confianza de los usuarios.

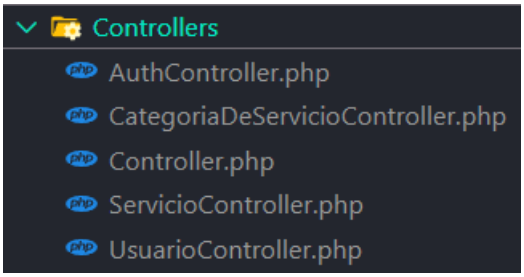
Integración y Beneficios

La combinación de PHP, Laravel y PostgreSQL proporciona una base tecnológica robusta y moderna para "Talento". PHP y Laravel trabajan en conjunto para ofrecer un desarrollo backend rápido y eficiente, mientras que PostgreSQL asegura una gestión de datos confiable y escalable. Esta arquitectura tecnológica permite que la plataforma maneje eficientemente las transacciones y comunicaciones entre proveedores y clientes, proporcionando una experiencia de usuario fluida y segura.

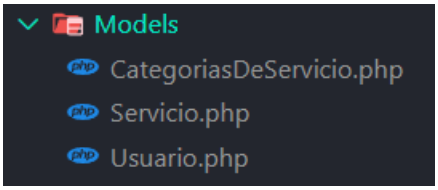
PHP se encarga de la lógica de negocio y el procesamiento de solicitudes, **Laravel** facilita el desarrollo estructurado y la integración de funcionalidades avanzadas, y **PostgreSQL** gestiona los datos con alta eficiencia y seguridad. La sinergia de estas tecnologías garantiza que "Talento" sea una solución efectiva para conectar a proveedores especializados con clientes, resolviendo de manera eficiente la problemática en el ámbito de la contratación de servicios profesionales.

NOMENCLATURAS DE NOMBRES UTILIZADOS PARA LOS DIFERENTES ELEMENTOS DEL PROYECTO

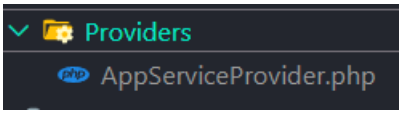
PascalCase: Cada palabra empieza con mayúscula y no se usan guiones ni guiones bajos.



Title Case/Capitalize: La primera letra de la palabra está en mayúscula y el resto en minúscula.



PascalCase: Cada palabra empieza con mayúscula y no se usan guiones ni guiones bajos.



• Explicación de los componentes principales (Modelos, Vistas, Controladores relevantes; si se repite el uso de alguno solamente utilizar el primero)

- **Modelos**

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class CategoriasDeServicio extends Model
{
    use HasFactory;

    protected $primaryKey = 'id_categoria';

    public function servicios()
    {
        return $this->hasMany(Servicio::class, 'id_categoria');
    }
}

```

Este modelo de categorías representa una categoría en la base de datos el cual utiliza el mapeo objeto-relacional el cual nos ayuda ha gestionar la interacción entre las tablas correspondientes, podemos observar que incluye dos atributos los cuales son nombre de categoría y la descripción de la misma la cuales podemos asignar masivamente gracias a la propiedad de $fillable y por ultimo este modelo esta relacionado con el modelo de servicio, esta relación es de uno a muchos.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class Usuario extends Authenticatable
{
    use HasFactory;

    protected $primaryKey = 'id_usuario';

    protected $fillable = ['nombre', 'email', 'password'];

    protected $hidden = ['password'];

    public function servicios()
    {
        return $this->belongsToMany(Servicio::class, 'servicios_personales', 'id_usuario', 'id_servicio')
                    ->withPivot('fecha_contratacion', 'estado_contratacion');
    }
}

```

Este modelo nos sirve para poder representar a un usuario en una base de datos, el cual extiende la clase Authenticatable la cual nos permite manejar la autenticación de usuario de manera integrada, en este modelo también se encuentra la propiedad $hidden la cual nos sirve para nos sirve para ocultar ciertos atributos tales como la contraseña, de igual forma contampos con la función de casts la cual ayuda a definir los atributos de email y contraseña ya que estos elemento se deben convertir en tipos específicos esto con el fin depoder facilitar la gestión de los datos dentro de la aplicación.

```php
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Servicio extends Model
{

    protected $table = 'servicios_personales';
    protected $primaryKey = 'id_servicios_personales';

    protected $fillable = [
        'id_categoria', 'id_usuario', 'nombre_servicio', 'descripcion_servicio', 'precio', 'imagen', 'numero_contacto',
    ];

    public function categoria()
    {
        return $this->belongsTo(CategoriasDeServicio::class, 'id_categoria');
    }

    public function usuario()
    {
        return $this->belongsTo(Usuario::class, 'id_usuario');
    }
}

```

Este modelo PHP, llamado Servicio, define la estructura y las relaciones de la tabla servicios_personales en una aplicación Laravel. Facilita las operaciones de base de datos y la interacción con otros modelos, como CategoriasDeServicio y Usuario.

- **Vistas**

```html
<head>
    <meta charset="utf-8">
    <title>Talento</title>
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
    <meta content="Free HTML Templates" name="keywords">
    <meta content="Free HTML Templates" name="description">

    <!-- Favicon -->
    <link rel="icon" type="image/png" href="img/favicon.png">

    <!-- Google Web Fonts -->
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=swap" rel="stylesheet">

    <!-- Font Awesome -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css" rel="stylesheet">

    <!-- Libraries Stylesheet -->
    <link href="lib/owlcarousel/assets/owl.carousel.min.css" rel="stylesheet">
    <link href="lib/tempusdominus/css/tempusdominus-bootstrap-4.min.css" rel="stylesheet" />

    <!-- Customized Bootstrap Stylesheet -->
    <link href="css/style.css" rel="stylesheet">
```

En esta vista se puede observar la información general de talento, algunas imágenes a forma de carrusel para poder enriquecer un poco más la página y

que así los usuarios puedan tener una información más clara de lo que se puede hacer dentro de la aplicación.

```
<body>
    <!-- Formulario -->
    <div class="container py-5">
        <h2>AGREGAR SERVICIO</h2>
        <form action="{{ route('servicio.store') }}" method="post">
            @csrf
            <div class="mb-3">
                <label for="nombre_servicio" class="form-label">Nombre</label>
                <input type="text" class="form-control" id="nombre_servicio" name="nombre_servicio">
            </div>
            <div class="mb-3 form-floating">
                <textarea class="form-control" placeholder="Leave a comment here" id="descripcion_servicio" style="height: 100px"
                    name="descripcion_servicio"></textarea>
                <label for="descripcion_servicio">Descripción</label>
            </div>
```

La página contiene un formulario que permite a los usuarios agregar un nuevo servicio a la plataforma. La estructura del formulario incluye campos para ingresar el nombre del servicio, una descripción, el precio, la categoría a la que pertenece y la URL de la imagen asociada al servicio. Además, se proporcionan botones para guardar la información ingresada o cancelar la operación y regresar a la página principal de servicios.

```
<body>
    <!-- Formulario -->
    <div class="container py-5">
        <h2>MODIFICAR SERVICIO</h2>
        <form action="{{ route('servicio.update', $servicio) }}" method="post">
            @csrf
            @method('PUT')
            <div class="mb-3">
                <label for="nombre_servicio" class="form-label">Nombre</label>
                <input type="text" class="form-control" id="nombre_servicio" name="nombre_servicio"
                    value="{{ $servicio->nombre_servicio }}">
            </div>
            <div class="mb-3 form-floating">
```

Esta vista contiene una listad e información la cual se ha llenado previamente a la hora que se agregar el servicio, pero nos sirve para editar alguno de los campos en el cual nos hayamos equivocado o para actualizar algún dato.

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Talento | Servicios</title>
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
    <meta content="Free HTML Templates" name="keywords">
    <meta content="Free HTML Templates" name="description">

    <!-- Favicon -->
    <link rel="icon" type="image/png" href="img/favicon.png">

    <!-- Google Web Fonts -->
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=swap" rel="stylesheet">

    <!-- Font Awesome -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css" rel="stylesheet">

    <!-- Libraries Stylesheet -->
    <link href="lib/owlcarousel/assets/owl.carousel.min.css" rel="stylesheet">
    <link href="lib/tempusdominus/css/tempusdominus-bootstrap-4.min.css" rel="stylesheet" />

    <!-- Customized Bootstrap Stylesheet -->
    <link href="css/style.css" rel="stylesheet">
</head>
```

La página comienza con un encabezado que incluye una barra de navegación superior con información de contacto y enlaces a redes sociales. La barra de navegación principal tiene un logotipo y enlaces a diferentes secciones del sitio, como "Inicio", "Nosotros", "Explorar", "Mis servicios" y "Guest"., esto con el fin de que los usuarios puedan explorar un poco más hay y poder conocer mejor quienes somos como talento.

- **Controladores.**

```php
<?php

namespace App\Http\Controllers;

use App\Models\Servicio;
use App\Models\CategoriasDeServicio;

use Illuminate\Http\Request;

class CategoriaDeServicioController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $categorias = CategoriasDeServicio::all();
        $servicios = Servicio::all();
        return view('index', compact('categorias','servicios'));
    }
```

Con este controlador podemos manejar las operaciones quede un CRUD, todas estas acciones para la entidad categoría se utilizan diferentes competentes los cuales nos ayudan a poder gestionar todas estas operaciones y que se puedan realizar. con éxito.

DESCRIPCIÓN DE LOS DIRECTORIOS PRINCIPALES DEL PROYECTO

Directorios Principales

- **app**: Contiene el código fuente de la aplicación.
    - **Http**: Controladores, middleware y otras clases relacionadas con las solicitudes HTTP.
        - **Controllers**: Controladores que manejan la lógica de las rutas.
    - **Models**: Modelos de Eloquent que representan las tablas de la base de datos.
        - **Categoria.php, Servicio.php, User.php**: Modelos específicos para la aplicación.
    - **Providers**: Proveedores de servicios que configuran la aplicación.
- **bootstrap**: Archivos de arranque y configuración de la aplicación.
    - **cache**: Archivos de caché generados por Laravel.
- **config**: Archivos de configuración de la aplicación.
    - **app.php, auth.php, backup.php, cache.php, database.php, filesystems.php, logging.php, mail.php, queue.php, services.php, session.php**: Configuraciones específicas para diferentes aspectos de la aplicación.
- **database**: Archivos relacionados con la base de datos.
    - **factories**: Definiciones de fábricas de Eloquent.
        - **UserFactory.php**: Fábrica para generar usuarios de prueba.
    - **migrations**: Archivos de migraciones de la base de datos.
    - **seeders**: Semillas para poblar la base de datos con datos iniciales.
        - **DatabaseSeeder.php**: Seeder principal de la base de datos.
    - **database.sqlite**: Archivo de base de datos SQLite.
- **public**: Contiene los archivos públicos accesibles desde el navegador.

        - **css, img, js, lib, scss**: Directorios para hojas de estilo, imágenes, JavaScript, librerías y archivos SCSS.
        - **index.php**: Punto de entrada de la aplicación.
    - **resources**: Archivos de recursos como vistas, CSS y JavaScript.
        - **views**: Plantillas Blade para las vistas.
            - **index.blade.php**: Plantilla de vista principal.
        - **css, js**: Archivos CSS y JavaScript del lado del cliente.
    - **routes**: Definiciones de rutas de la aplicación.
        - **web.php**: Rutas web de la aplicación.
        - **console.php**: Rutas para comandos de la consola.
    - **storage**: Archivos generados por la aplicación.
        - **app**: Almacenamiento de archivos de la aplicación.
        - **framework**: Caché, sesiones y vistas compiladas.
        - **logs**: Archivos de registro.
    - **tests**: Pruebas automatizadas para la aplicación.
        - **Feature**: Pruebas de características.
            - **ExampleTest.php**: Ejemplo de prueba de característica.
        - **Unit**: Pruebas unitarias.
            - **ExampleTest.php**: Ejemplo de prueba unitaria.
        - **TestCase.php**: Clase base para todas las pruebas.
    - **vendor**: Dependencias del proyecto gestionadas por Composer.

Archivos Principales

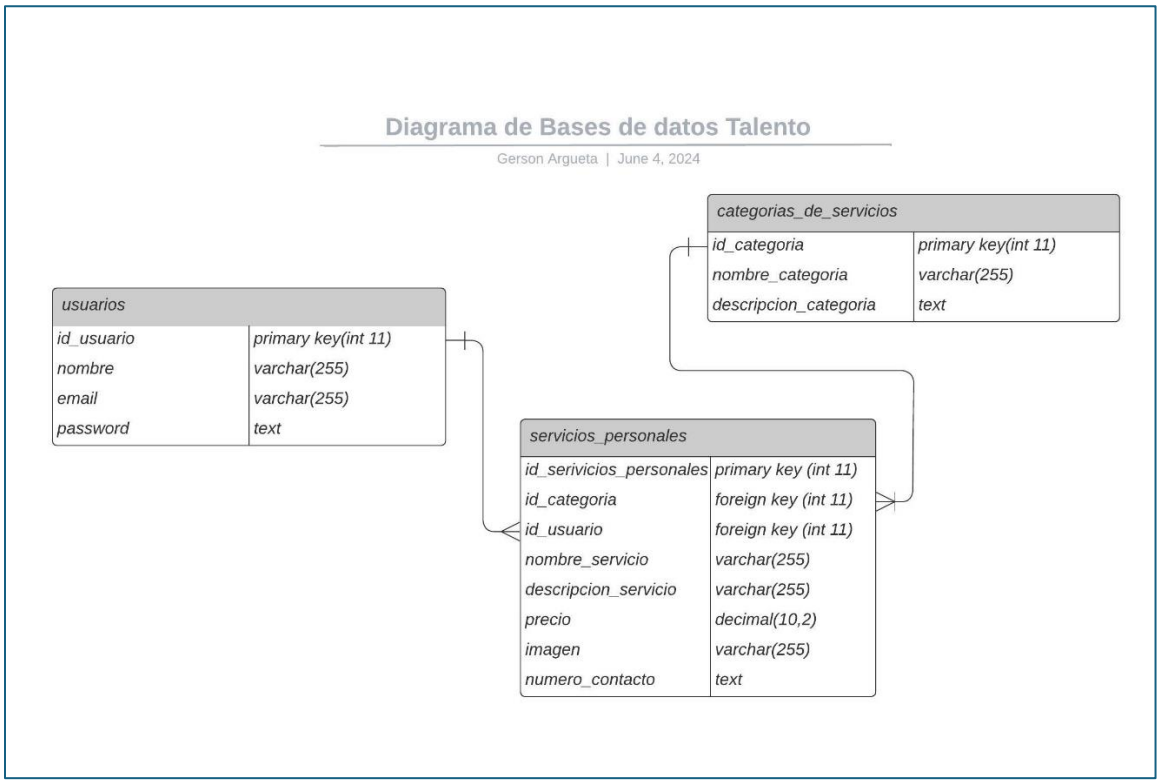
- **.editorconfig**: Configuración del editor de texto.
- **.env**: Variables de entorno para configuración de la aplicación.
- **.env.example**: Ejemplo de variables de entorno.

- **.gitattributes**: Atributos de Git.
- **.gitignore**: Archivos y directorios que Git debe ignorar.
- **artisan**: Ejecutable de la consola de Laravel.
- **composer.json**: Configuración de dependencias de Composer.
- **composer.lock**: Archivo de bloqueo de versiones de dependencias.
- **package.json**: Configuración de dependencias de Node.js.
- **phpunit.xml**: Configuración de PHPUnit para pruebas.
- **README.md**: Documentación del proyecto.

Resumen Breve de Funcionalidades de Carpetas Clave

- **app**: Código fuente principal y lógica de la aplicación.
- **bootstrap**: Configuración de arranque.
- **config**: Archivos de configuración.
- **database**: Migraciones, fábricas y seeds.
- **public**: Archivos accesibles públicamente.
- **resources**: Vistas y recursos de frontend.
- **routes**: Definición de rutas.
- **storage**: Almacenamiento de archivos generados.
- **tests**: Pruebas automatizadas.
- **vendor**: Dependencias externas gestionadas por Composer.

DIAGRAMA ENTIDAD-RELACIÓN DE LA BASE DE DATOS



Relación entre Tablas

1. **Tabla servicios_personales:**

Relaciones:

- **Con usuarios**:

  La columna id_usuario en servicios_personales es una clave foránea que referencia a la columna id_usuario en la tabla usuarios. Esto establece una relación de uno a muchos, donde un usuario puede tener varios registros en servicios_personales.

- **Con categorias_de_servicios**:

  La columna id_categoria en servicios_personales es una clave foránea que referencia a la columna id_categoria en la tabla categorias_de_servicios. Esto establece una relación de uno a

muchos, donde una categoría puede tener varios registros en servicios_personales.

2. **Tabla categorias_de_servicios:**

    - **Relación con servicios_personales:**

La tabla categorias_de_servicios está relacionada con la tabla servicios_personales mediante la columna id_categoria, que es una clave primaria en categorias_de_servicios y una clave foránea en servicios_personales. Esto significa que cada servicio personal pertenece a una categoría específica. La relación es de uno a muchos (una categoría puede tener muchos servicios personales).

3. **Tabla usuarios:**

    - **Relación con servicios_personales:**

La tabla usuarios tiene una relación con la tabla servicios_personales a través de la columna id_usuario, que es una clave primaria en usuarios y una clave foránea en servicios_personales. Esto indica que cada usuario puede tener múltiples servicios personales registrados en la tabla servicios_personales. La relación es de uno a muchos (un usuario puede tener muchos servicios personales).

EXPLICACIÓN DEL PROPÓSITO DE CADA RUTA

```
PS C:\Users\jeffe\OneDrive\Documentos\TalentoFinal\talento> php artisan route:list

  GET|HEAD        # ..... #inicio
  GET|HEAD        #educacion ..... educacion
  GET|HEAD        #explorar ..... todos
  GET|HEAD        #leyes ..... leyes
  GET|HEAD        #negocios ..... negocios
  GET|HEAD        #nosotros ..... nosotros
  GET|HEAD        #tech ..... tech
  GET|HEAD        #testimonios ..... testimonios
  GET|HEAD        / ..... inicio › CategoriaDeServicioController@index
  POST            _ignition/execute-solution ..... ignition.executeSolution › Spatie\LaravelIgnition › ExecuteSolutionController
  GET|HEAD        _ignition/health-check ..... ignition.healthCheck › Spatie\LaravelIgnition › HealthCheckController
  POST            _ignition/update-config ..... ignition.updateConfig › Spatie\LaravelIgnition › UpdateConfigController
  GET|HEAD        educación ..... categoriaEducacion › CategoriaDeServicioController@educacion
  GET|HEAD        informacion{servicio} ..... servicio.info › ServicioController@info
  POST            iniciar ..... iniciar › AuthController@login
  GET|HEAD        legales ..... categoriaLeyes › CategoriaDeServicioController@leyes
  GET|HEAD        login ..... login › AuthController@showLoginForm
  GET|HEAD        logout ..... logout › AuthController@logout
  GET|HEAD        negocios ..... categoriaNegocios › CategoriaDeServicioController@negocios
  GET|HEAD        register ..... register › AuthController@showRegistrationForm
  POST            registrarse ..... registrarse › AuthController@register
  GET|HEAD        servicio ..... servicio.index › ServicioController@index
  POST            servicio ..... servicio.store › ServicioController@store
  GET|HEAD        servicio/create ..... servicio.create › ServicioController@create
  GET|HEAD        servicio/{servicio} ..... servicio.show › ServicioController@show
  PUT|PATCH       servicio/{servicio} ..... servicio.update › ServicioController@update
  DELETE          servicio/{servicio} ..... servicio.destroy › ServicioController@destroy
  GET|HEAD        servicio/{servicio}/edit ..... servicio.edit › ServicioController@edit
  GET|HEAD        servicios ..... category › ServicioController@category
  GET|HEAD        tecnología ..... categoriaTech › CategoriaDeServicioController@tech
  GET|HEAD        up .....
```

php artisan route:list (Ver rutas disponibles)

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ServicioController;
use App\Http\Controllers\CategoriaDeServicioController;
use App\Http\Controllers\AuthController;
use App\Http\Middleware\AuthM;


// Navb routes
Route::get('/', [CategoriaDeServicioController::class, 'index'])->name('inicio');
Route::get('/tecnología', [CategoriaDeServicioController::class, 'tech'])->name('categoriaTech');
Route::get('/legales', [CategoriaDeServicioController::class, 'leyes'])->name('categoriaLeyes');
Route::get('/educación', [CategoriaDeServicioController::class, 'educacion'])->name('categoriaEducacion');
Route::get('/negocios', [CategoriaDeServicioController::class, 'negocios'])->name('categoriaNegocios');

Route::get('servicios', [ServicioController::class, 'category'])->name('category');
Route::get('informacion{servicio}', [ServicioController::class, 'info'])->name('servicio.info');



Route::get('/#')->name('#inicio');
Route::get('/#nosotros')->name('nosotros');
Route::get('/#testimonios')->name('testimonios');
Route::get('/#explorar')->name('todos');
Route::get('/#tech')->name('tech');
Route::get('/#negocios')->name('negocios');
Route::get('/#educacion')->name('educacion');
Route::get('/#leyes')->name('leyes');


//Route::get('/')
Route::middleware(AuthM::class)->group(function(){
    Route::resource('servicio', ServicioController::class);

});

// Routes auth
Route::get('register', [AuthController::class, 'showRegistrationForm'])->name('register');
Route::post('registrarse', [AuthController::class, 'register'])->name('registrarse');
Route::get('login', [AuthController::class, 'showLoginForm'])->name('login');
Route::post('iniciar', [AuthController::class, 'login'])->name('iniciar');
Route::get('logout', [AuthController::class, 'logout'])->name('logout');


```

Web.php (Rutas web)

Utilidad de cada ruta:

1. **Inicio y categorías principales:**
    - **GET /:** Muestra la página de inicio (index del CategoriaDeServicioController). Nombre de la ruta: inicio.
    - **GET /tecnología:** Muestra la categoría de tecnología (tech del CategoriaDeServicioController). Nombre de la ruta: categoriaTech.
    - **GET /legales:** Muestra la categoría de legales (leyes del CategoriaDeServicioController). Nombre de la ruta: categoriaLeyes.
    - **GET /educación:** Muestra la categoría de educación (educacion del CategoriaDeServicioController). Nombre de la ruta: categoriaEducacion.
    - **GET /negocios:** Muestra la categoría de negocios (negocios del CategoriaDeServicioController). Nombre de la ruta: categoriaNegocios.
2. **Servicios:**
    - **GET /servicios:** Muestra la lista de todas las categorías de servicios (category del ServicioController). Nombre de la ruta: category.
    - **GET /informacion{servicio}:** Muestra la información detallada de un servicio específico (info del ServicioController). Nombre de la ruta: servicio.info.

3. **Secciones del navbar (ancladas con #):**
    - **GET /#:** Ancla para la sección de inicio. Nombre de la ruta: #inicio.
    - **GET /#nosotros:** Ancla para la sección "nosotros". Nombre de la ruta: nosotros.
    - **GET /#testimonios:** Ancla para la sección "testimonios". Nombre de la ruta: testimonios.
    - **GET /#explorar:** Ancla para la sección "explorar todos". Nombre de la ruta: todos.
    - **GET /#tech:** Ancla para la sección "tech". Nombre de la ruta: tech.
    - **GET /#negocios:** Ancla para la sección "negocios". Nombre de la ruta: negocios.
    - **GET /#educacion:** Ancla para la sección "educación". Nombre de la ruta: educacion.
    - **GET /#leyes:** Ancla para la sección "legales". Nombre de la ruta: leyes.
4. **CRUD de servicios (protegido por middleware de autenticación AuthM):**
    - **Route::resource('servicio', ServicioController::class):** Define todas las rutas necesarias para CRUD de servicios:
        - **GET /servicio:** Lista todos los servicios (index).
        - **POST /servicio:** Guarda un nuevo servicio (store).
        - **GET /servicio/create:** Muestra el formulario para crear un nuevo servicio (create).

- **GET /servicio/{servicio}:** Muestra un servicio específico (show).
- **PUT/PATCH /servicio/{servicio}:** Actualiza un servicio específico (update).
- **DELETE /servicio/{servicio}:** Elimina un servicio específico (destroy).
- **GET /servicio/{servicio}/edit:** Muestra el formulario para editar un servicio (edit).

5. **Autenticación:**
    - **GET /register:** Muestra el formulario de registro (showRegistrationForm). Nombre de la ruta: register.
    - **POST /registrarse:** Procesa el registro de un nuevo usuario (register). Nombre de la ruta: registrarse.
    - **GET /login:** Muestra el formulario de inicio de sesión (showLoginForm). Nombre de la ruta: login.
    - **POST /iniciar:** Procesa el inicio de sesión (login). Nombre de la ruta: iniciar.
    - **GET /logout:** Cierra la sesión del usuario (logout). Nombre de la ruta: logout.

Rutas adicionales del listado (php artisan route:list)

Estas rutas adicionales están relacionadas con el paquete **Spatie\LaravelIgnition:**

- **Ejecutar soluciones de ignition:**
    - **POST _ignition/execute-solution:** Ejecuta una solución propuesta por Ignition (ExecuteSolutionController).
    - **GET|HEAD _ignition/health-check:** Verifica la salud del sistema (HealthCheckController).
    - **POST _ignition/update-config:** Actualiza la configuración de Ignition (UpdateConfigController).