# Documentación StellarHost

## Análisis de requerimientos

El cliente entrego los siguientes requerimientos:

* Sitio web acorde al diseño (se anexa diseño en photoshop)
* Implementar estrategia de búsqueda para los servicios que pueda ofrecer la compañía.
* Implementar el mantenimiento de datos (altas, bajas y actualizaciones de los registros orientado a los servicios que se ofrecen en el hosting (solo para administradores).
* Los administradores tendrán su forma de autenticación, por lo cual el sitio tendrá actividad pública y privada.
* Deberá ser responsivo
* Deberá proporcionar el mapa de ubicación geográfica, el punto físico de la empresa (View map)
* Generar un video donde explique el objetivo de la página/empresa y cuál es la finalidad, alguna promoción etcétera o consejos, video de 2 a 4 minutos, y claro, ubicarlo en la sección o link que usted crea conveniente en el sitio web.
* Implementar un slideshow en la sección marcada en el Photoshop.
* Deberá contar con un formulario de contacto por correo electrónico.
* Deberá contar con un formulario de sugerencias para la empresa (no correo, tabla interna del sistema), con control captcha.
* Poder ver ese listado de sugerencias con paginación de sugerencias y exportarlos a Excel y pdf.
* En la sección clientes, colocar imágenes con respectivo link de compañías que hipotéticamente son tus clientes.
* Poder generar la búsqueda y un pdf con la ficha técnica de los datos de cada plan de hosting ofertado.
* Montar un tablero de noticias de forma dinámica, por base de datos.
* Implementar contador de visitas y usuarios conectados.
* En la sección de preguntas frecuentes utilizar un control tipo acordeón.
* Obtener 2 dashboard (con gráfico), el primero del análisis de los tipos de planes registrados y el segundo de los planes a vencer en los últimos 10 días.
* Implementar un chat bot para la ayuda en línea las 24 horas
* Cuidar la compatibilidad con los navegadores más importantes.
* Implementar la base de datos correspondiente.

## Especificación

## Diseño y arquitectura

## En el proyecto utilizamos Laravel (framework de PHP), utilizando el patrón MVC.

## Este patrón nos permite separar la lógica de la aplicación de la presentación y el manejo de datos.

## Laravel nos ofrece una serie de herramientas y funcionalidades que facilitan el desarrollo web, como el sistema de rutas, la autenticación, la validación, las migraciones, los seeders, etc.

## Además, Laravel se integra fácilmente con otros servicios y librerías, como Vue.js, Bootstrap, Tailwind CSS, etc. Laravel es un framework muy popular y robusto que nos ayuda a crear aplicaciones web de calidad y con buenas prácticas.

Para la base de datos utilizamos MySQL, un sistema de gestión de bases de datos relacionales que permite almacenar y manipular datos de forma eficiente y segura. MySQL es uno de los sistemas más populares y utilizados en el mundo, debido a su facilidad de uso, su rendimiento y su compatibilidad con diversos lenguajes de programación y plataformas.

## Programación

Dado que utilizamos MVC, para todo utilizamos un modelo para cada clase:

class registros extends Model

{

    use HasFactory;

    public function servicios()

    {

        return $this->belongsTo(servicios::class, 'id\_servicio');;

    }

    public function cliente()

    {

        return $this->belongsTo(clientes::class, 'id\_cliente');

    }

}

**Un Controlador**: Con distintos métodos, algunos de ellos son accedidos solamente por los administradores:

class registrosController extends Controller

{

    public function index()

    {

        $registros = registros::all();

        $clientes = clientes::all();

        $servicios = servicios::all();

        return view('admin.registros.index', ['registros' => $registros, 'clientes' => $clientes, 'servicios' => $servicios]);

    }

    public function store(Request $request)

    {

        $request->validate([

            'id\_servicio' => 'required|numeric',

            'id\_cliente' => 'required|numeric',

        ]);

        $registros = new registros;

        $fecha = Carbon::now();

        $registros->fecha\_contrato = $fecha;

        $addMonth = $fecha->copy()->addMonth(1);

        $registros->fecha\_instalacion = $addMonth;

        $registros->id\_servicio = $request->id\_servicio;

        $registros->id\_cliente = $request->id\_cliente;

        $registros->id = $request->id\_cliente . $request->id\_servicio . preg\_replace('([^0-9])', '', $fecha);

        $registros->save();

        return redirect()->route('registros.index')->with('success', 'Registro Exitoso');

    }

…

Todos los controladores contienen varias funciones por defecto:

**Index:** Muestra todos los registros del modelo.

**Store:** Ingresa un registro al modelo.

**Find:** Busca el dato que contenga el id que requiere la función.

**Update:** Actualiza el registro conforme al id asignado.

**Destroy:** Elimina el registro conforme al id asignado.

**Y su respectiva vista:**

Esta vista puede tener los siguientes métodos de Laravel 10:

@extends: Extrae el contenido de una vista especificada.

@section: El contenido cambia dependiendo de la vista, para esto hay que declarar @extends(‘[vista]’)

**Ruteo:**

En el archivo web.php declaramos el ruteo de nuestras vistas, métodos, get, post, etc.

//VIEWS

Route::view('/', 'users.template');

Route::view('login', 'admin.login')->name('login')->middleware('guest');

Route::view('tecnologia', 'users.tecnologia')->name('tecnologia');

Route::view('resellers', 'users.resellers')->name('resellers');

Route::view('FAQ', 'users.FAQ')->name('FAQ');

Route::view('todo', 'todo');

Route::view('usrservicios', 'users.servicios');

Route::view('client', 'users.client');

Route::view('hire', 'users.hire')->name('hire');

Route::view('success', 'users.success')->name('success');

Route::view('export', 'users.export')->name('export');

Route::view('contacto', 'users.contacto')->name('contacto');

Route::view('centro\_de\_ayuda', 'users.centro\_de\_ayuda')->name('centro\_de\_ayuda');

//RESOURCES

Route::resource('clientes', clientesController::class)->middleware('auth');

Route::resource('servicios', serviciosController::class)->middleware('auth');

//GET

Route::get('dashboard', [registrosController::class, 'mostrarRegistros'])->name('dashboard')->middleware('auth');

Route::get('todo', [noticiasController::class, 'indexGuest']);

Route::get('/', [serviciosController::class, 'indexall']);

//POST

Route::post('login', [loginController::class, 'login']);

Route::post('logout', [loginController::class, 'logout']);

Route::post('clientRegister', [clientesController::class, 'register'])->name('clientRegister');

Route::post('hireService', [registrosController::class, 'hireService'])->name('hireService');

## Pruebas

Dado que el tiempo no nos permitió trabajar con comodidad, las pruebas automáticas fueron omitidas. Todas y cada una de las pruebas que los desarrolladores realizaron fue mediante ejecución de la aplicación.

Distintos errores fueron arreglados buscando en la red, en páginas como stackoverflow, documentación de Laravel, documentación de dependencias (composer), OpenAI ChatGPT y Bing GPT-4

## Documentación

El cliente nos hizo entrega de dos documentos.

Uno con los requerimientos solicitados y otro en formato psd. El cual incluía el diseño a implementar en todas las páginas de usuario.

## Implementación

## Mantenimiento