Índice

1. Resumen ejecutivo

Este proyecto es una **aplicación web informativa y de contacto** para la práctica profesional del Dr. Jorge Luis Serna (veterinaria enfocada en reproducción animal). El sistema incluye: landing/portada, servicios, sobre mí, consultas virtuales (formulario que guarda en BD), blog dinámico, contacto avanzado, testimonios, casos de éxito y FAQ. Se implementa con **PHP** en backend, **MySQL** como base de datos y el frontend originalmente en Bootstrap, migrado a **Tailwind CSS**. Corre en entorno local Laragon (Windows).

El documento justifica cada elección técnica y de diseño, cubre la arquitectura, seguridad y pruebas para que el proyecto sea reproducible, mantenible y escalable.

2. Objetivos del sistema

- Proveer una presencia web profesional para servicios de reproducción animal.
- Permitir agendamiento de consultas (presencial/virtual) y contacto rápido.
- Mostrar información técnica (servicios, formación, casos de éxito) y publicar artículos (blog).
- Ser responsivo, accesible y fácil de mantener por desarrolladores con nivel intermedio.
- Centrar el diseño en la identidad visual solicitada: verde (#198754), blanco y negro.

3. Alcance funcional

Módulos implementados:

- Landing principal con hero, servicios, testimonios, resultados, FAQ y contacto rápido.
- Páginas estáticas: Servicios, Sobre mí.
- Formulario de Consultas Virtuales (guarda en consultas en BD).
- Formulario de Contacto (guarda en contacto en BD + enlace a WhatsApp).
- Blog dinámico: listados y vista individual (tabla blog).
- Backend minimal para insertar consultas y contactos.
- Sistema pensado para futura extensión (panel administrativo, subida de imágenes, exportes PDF/Excel).

4. Decisiones tecnológicas (justificación)

4.1 Lenguaje backend: PHP

- **Justificación**: Entorno educativo (Laragon) y requerimiento inicial del usuario; facilidad para desplegar en servidor LAMP/Windows; integración directa con MySQL sin capas complejas.
- **Ventajas**: Bajo coste de setup en Laragon, amplia documentación, facilidad para principiantes y compatibilidad con hosting compartido si se despliega.

4.2 Base de datos: MySQL

- **Justificación**: Relacional, suficiente para las tablas del proyecto (contacto, consultas, blog). Fácil de manejar con phpMyAdmin/HeidiSQL en Laragon.
- **Ventajas**: ACID, simple modelado para consultas y escalable a medida que se añadan tablas (usuarios, roles, historial).

4.3 Frontend: Tailwind CSS (migración desde Bootstrap)

- **Justificación**: Tailwind brinda control atómico, menor CSS personalizado si se configura correctamente y permite un estilo coherente con la paleta del proyecto. Atención a la petición del usuario: migrar a Tailwind.
- **Ventajas**: Consistencia, fácil tematización, peso reducido tras build/optimización (purge/Content), y mayor control de estilos sin crear hojas CSS grandes.

4.4 Entorno de desarrollo: Laragon + Node.js (Tailwind build)

- **Justificación**: Laragon es ideal para Windows; Node.js es necesario para el pipeline de Tailwind (build/watch).
- **Ventajas**: Desarrollo local controlado, comandos simples npm run build y npm run watch.

4.5 Librerías complementarias

- **Lucide / Heroicons**: iconos simples, ligeros y escalables para UI moderna.
- **Google Fonts (Poppins)**: tipografía moderna y legible.

5. Arquitectura global

5.1 Modelo general

- **Cliente (Navegador)**: HTML/PHP renderizado en servidor, CSS Tailwind, JS mínimo para interacciones (toggle nav, acordeón).
- **Servidor Web**: Apache (Laragon) procesa archivos PHP.
- **DB**: MySQL, conexiones mediante mysqli (o PDO si se decide mejorar).
- **Assets**: assets/css/tailwind.css, assets/img/, assets/js/script.js (si se requiere).

5.2 Flujos principales

- Usuario envía formulario → backend/*.php recibe POST → sanitiza y guarda en BD → responde con redirección/alert.

- Visualización blog: blog.php consulta tabla blog y renderiza listado; ver_articulo.php?id=X muestra detalle.

6. Estructura de ficheros y responsabilidades

Directorio raíz: veterinaria/

veterinaria/

├─ index.php # Landing (hero, servicios, testimonios, casos de éxito, FAQ, contacto rápido)

├─ header.php # Cabecera (navbar) y cargado de assets

├─ footer.php # Footer

├─ servicios.php # Página servicios (detallada)

├─ sobre.php # Sobre el doctor

├─ consultas.php # Formulario de consultas virtuales

├─ blog.php # Listado de artículos

├─ ver_articulo.php # Ver artículo individual

├─ contacto.php # Página de contacto avanzada

├─ agendar.php # Página de agendamiento (si aplica)

├─ backend/

│ ├─ agendar_consulta.php # Inserta en tabla consultas

│ ├─ guardar_contacto.php # Inserta en tabla contacto

│ └─ agregar_articulo.php # Script temporal para insertar artículos

├─ config/

│ └─ db.php # Conexión a MySQL (mysqli)

├─ assets/

│ ├─ css/

│ │ └─ tailwind.css # Generado por Tailwind build

│ ├─ img/ # Imágenes (doctor.jpg, iatf.jpg, etc.)

│ └─ js/

│ └─ script.js # JS global (nav toggle, acordeón, etc.)

```
├─ src/
│   └─ styles/
│       └─ tailwind.css          # Input de Tailwind (@tailwind base; ...)
├─ database/
│   └─ veterinaria.sql           # DDL para crear BD y tablas
└─ package.json                  # Dependencias Node (tailwind, postcss)
```

Responsabilidades clave:

- header.php y footer.php: importación única, evita duplicación, facilita migraciones.
- config/db.php: centraliza credenciales de BD.
- backend/: scripts que manejan inserciones en BD y lógica POST.
- assets/css/tailwind.css: único CSS compilado, evita carga de frameworks redundantes.

7. Base de datos — esquema y decisiones (DDL)

7.1 Reglas de diseño

- Normalización simple (1NF/2NF) para evitar duplicidad.
- Campos básicos con tamaños adecuados, uso de TIMESTAMP para auditar.
- Seguridad: uso de real_escape_string en PHP (recomendado: migrar a prepared statements / PDO para producción).

7.2 DDL (completo)

```
CREATE DATABASE IF NOT EXISTS veterinaria;

USE veterinaria;


-- Tabla contactos (formularios de contacto)

CREATE TABLE IF NOT EXISTS contacto (

  id INT AUTO_INCREMENT PRIMARY KEY,

  nombre VARCHAR(100) NOT NULL,

  telefono VARCHAR(20),
```

```
  correo VARCHAR(100),
  servicio VARCHAR(100),
  especie VARCHAR(50),
  mensaje TEXT,
  fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabla consultas virtuales
CREATE TABLE IF NOT EXISTS consultas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  correo VARCHAR(100),
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  mensaje TEXT,
  fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabla blog (artículos)
CREATE TABLE IF NOT EXISTS blog (
  id INT AUTO_INCREMENT PRIMARY KEY,
  titulo VARCHAR(255) NOT NULL,
  contenido TEXT NOT NULL,
  fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  imagen VARCHAR(255) DEFAULT NULL
);
```

7.3 Índices y rendimiento

- Índice por fecha en blog implícito por el orden de inserción (si hay mucha lectura, crear índice INDEX(fecha)).

- Para búsquedas por servicio/especie en contacto, crear índices si se requiere filtrado frecuente.

8. Diseño de interfaz — paleta, tipografía, accesibilidad

8.1 Paleta de colores (oficial)

- --verde-principal: #198754 (primary)
- --verde-claro: #20C997 (primaryLight)
- --blanco: #FFFFFF
- --negro: #000000
- --gris-oscuro: #343A40
- --gris-claro: #F8F9FA

Uso

- Verde principal: CTAs (botones primarios), títulos, acentos.
- Blanco: fondo general y botones secundarios sobre verde.
- Negro: textos fuertes, header/nav.
- Grises: texto secundario y fondos de secciones.

8.2 Tipografía

- **Poppins**: usada en títulos y cuerpo.
- Pesos: 300, 400, 600, 700.
- Escala propuesta:
    - H1 hero: 48–64px (clase text-5xl/text-6xl en Tailwind).
    - H2 secciones: 32px (text-3xl).
    - Texto base: 16px (text-base).
    - Pequeño: 14px (text-sm).

8.3 Iconografía

- **Lucide** o **Heroicons**: SVGs livianos y semánticos.
- Consistencia: todos lineales (stroke) o sólidos — elegir un estilo.

8.4 Accesibilidad (a11y)

- Contraste: verde sobre blanco y blanco sobre verde deben superar ratio 4.5:1 (verificar con herramientas).
- Etiquetas label en inputs; aria-expanded en acordeones; alt en imágenes.
- Navegación por teclado: focus visible (focus:ring en Tailwind).
- Tamaño de hit-target: botones >= 44x44 px táctil.
- Semántica HTML: usar <header>, <main>, <section>, <footer> cuando se mejore la plantilla.

---

9. Estándares de código y convenciones de nombres

9.1 PHP

- Archivos en minúsculas snake_case.php (ej: guardar_contacto.php).
- Variables $conn, $sql y $row para consistencia.
- Documentar funciones con comentarios si se añaden.

9.2 CSS / Tailwind

- Clases utilitarias de Tailwind; mínimo CSS custom.
- Colores extendidos en tailwind.config.js (primary, primaryLight).
- Evitar estilos inline duplicados; si se repiten, crear @layer components en el src/styles/tailwind.css.

9.3 JS

- Archivo assets/js/script.js para funciones globales (nav toggle, acordeón). Evitar código inline extenso.

10. Seguridad

10.1 Conexión BD

- config/db.php usa mysqli. **Recomendación**: migrar a PDO y prepared statements para prevenir SQL Injection.
- Guardar credenciales en archivo config/db.php; en producción, usar variables de entorno.

10.2 Validación y sanitización

- En backend, usar:

    - $conn->real_escape_string() (temporal) **pero** preferir prepared statements.
    - Validación del lado cliente (HTML5) para UX y validación servidor para seguridad.
- Validación de tipo/longitud y escapado al mostrar (htmlspecialchars) si se muestra contenido de usuarios en páginas.

10.3 CSRF y XSS

- Añadir tokens CSRF en formularios para producción (por ejemplo, generar token en sesión).
- Escapar salida para prevenir XSS: echo htmlspecialchars($texto, ENT_QUOTES, 'UTF-8').

10.4 Restricción de subida de archivos

- Si se permite subir imágenes (blog), validar:
    - Tipo MIME y extensión.
    - Tamaño límite (p. ej. 2MB).
    - Almacenamiento en carpeta con permisos seguros y nombre único (hash).

11. Migración Bootstrap → Tailwind (plan técnico paso a paso)

Objetivo

Sustituir por completo el frontend Bootstrap por Tailwind manteniendo toda la funcionalidad y estructura del sitio.

11.1 Resumen de pasos (ejecución)

1. **Backup completo** del proyecto.
2. **Instalación Node.js** y inicializar npm en la raíz del proyecto.
3. **Instalar Tailwind**: npm install -D tailwindcss postcss autoprefixer y npx tailwindcss init -p.
4. **Crear src/styles/tailwind.css** con @tailwind base; @tailwind components; @tailwind utilities;.
5. **Configurar tailwind.config.js** para que lea .php (content: ["./*.php", "./**/*.php", "./src/**/*.{js,css}"]) y extender colores primary/primaryLight.
6. **Agregar scripts** en package.json: build y watch. Ejecutar npm run build.

7. **Reemplazar en header.php** la inclusión de bootstrap por assets/css/tailwind.css. Mantener fuentes e iconos.
8. **Migrar header.php, footer.php, index.php** a Tailwind (te entregué versiones completas). Testear.
9. **Migrar página por página**: servicios.php, consultas.php, contacto.php, blog.php, ver_articulo.php. Probar envío de formularios.
10. **Sustituir JS dependiente de Bootstrap** (por ej. acordeón, collapse) por implementaciones ligeras (vanilla JS o Alpine.js).
11. **Eliminar referencias a Bootstrap** cuando todo esté probado.
12. **Ejecutar npm run build --production** (o --minify) para archivo final optimizado.

11.2 Reglas de conversión (mapeos)

- container → max-w-7xl mx-auto px-4.
- row + col-md-* → flex flex-wrap -mx-4 + children px-4 w-full md:w-1/2 lg:w-1/3.
- card → bg-white shadow rounded-lg p-4.
- btn btn-success → bg-primary text-white px-4 py-2 rounded-md.
- text-center, mt-3 → se mantienen como utilidades text-center mt-3.

11.3 Recomendación de herramientas

- VSCode + extensión *Search & Replace*, para revisar cada class="..." y convertir manualmente.
- npm run watch durante la migración para ver cambios en tiempo real.

12. Integración y despliegue en Laragon (instrucciones exactas)

12.1 Requisitos

- Laragon instalado (incluye Apache, MySQL, phpMyAdmin).
- Node.js LTS instalado.
- PHP 7.4+ recomendado.

12.2 Pasos

1. Coloca la carpeta veterinaria en C:\laragon\www\.
2. Inicia Laragon (Start All).

3. Crea base de datos:
    - Abre http://localhost/phpmyadmin → crear BD veterinaria.
    - Importa database/veterinaria.sql.
4. Configura config/db.php con tus credenciales MySQL (usuario, contraseña). Ejemplo:

```
<?php
$host = "localhost";
$user = "Parra";
$pass = "Colombia2024*";
$db   = "veterinaria";

$conn = new mysqli($host, $user, $pass, $db);
if ($conn->connect_error) {
   die("Error de conexión a la base de datos: " . $conn->connect_error);
}
?>
```

5. Instala dependencias Node y genera CSS:

```
cd C:\laragon\www\veterinaria
npm install
npm run build        # genera assets/css/tailwind.css
# durante desarrollo:
npm run watch
```

6. Abre en navegador: http://localhost/veterinaria/.

13. Pruebas y validación (QA)

13.1 Pruebas funcionales

- Envío de formulario consultas.php: registro correcto en tabla consultas.
- Envío contacto.php: registro en contacto.
- Visualización blog.php y ver_articulo.php.

- Links del navbar funcionan.
- Menú móvil (toggle) abre/cierra correctamente.
- Acordeón FAQ abre y cierra.

13.2 Pruebas de compatibilidad

- Chrome Desktop, Firefox, Edge.
- Modo móvil via DevTools (iPhone/Android sizes).
- Resoluciones: 1440px, 1024px, 768px, 375px.

13.3 Pruebas de seguridad básicas

- Intentar enviar caracteres especiales en inputs y verificar que no rompan consultas.
- Prueba básica de inyección simple (sin prepared statements, algunas inyecciones pueden fallar) — **importante**: migrar a prepared statements.

13.4 Accesibilidad

- Lighthouse audit (Chrome) — objetivo: puntaje >= 90 en accesibilidad.
- Comprobar contraste de colores y tab-navigability.

14. Operación y mantenimiento

14.1 Backups

- Backup MySQL: programar exportación de BD .sql semanal.
- Backup de assets/img y backend en repositorio o copia en otro drive.

14.2 Logs

- Habilitar logs de Apache/PHP en Laragon para errores.
- Guardar errores críticos en archivo logs/php_errors.log (configurable en php.ini).

14.3 Actualizaciones

- Actualizar dependencias Node (Tailwind) cada 6 meses.
- Mantener PHP actualizado a versiones seguras.

15. Checklist final para entrega y evaluación

- Carpeta veterinaria/ completa con archivos PHP migrados a Tailwind.
- assets/css/tailwind.css generado (run build).
- BD veterinaria creada e importada.
- config/db.php con credenciales correctas (verificado).
- Formularios (contacto, consultas) insertan correctamente en BD.
- Blog muestra artículos y vista detalle.
- Navbar funcional en desktop y móvil.
- FAQ funciona con acordeón JS.
- Accesibilidad básica validada (contraste, focus).
- Documentación (este archivo) incluida en la entrega.
- Backup del proyecto (carpeta veterinaria-backup) incluido.

16. Anexos: comandos, ejemplos y fragmentos

16.1 Comandos esenciales

```
# Inicializar npm
npm init -y

# Instalar Tailwind y herramientas
npm install -D tailwindcss postcss autoprefixer

# Crear archivos de configuración (crea tailwind.config.js y postcss.config.js)
npx tailwindcss init -p

# Compilar CSS (producción / minificado)
npm run build

# Modo desarrollo (recompila automático)
```

```
npm run watch


# Exportar base de datos (ejemplo con mysqldump)
mysqldump -u Parra -p veterinaria > veterinaria_backup.sql
```

16.2 Ejemplo tailwind.config.js recomendado

```
module.exports = {
  content: [
    "./*.php",
    "./**/*.php",
    "./src/**/*.{html,js,css}"
  ],
  theme: {
    extend: {
      colors: {
        primary: "#198754",
        primaryLight: "#20C997"
      }
    }
  },
  plugins: []
}
```

16.3 Script package.json (fragmento)

```
"scripts": {
  "build": "tailwindcss -i ./src/styles/tailwind.css -o ./assets/css/tailwind.css --minify",
  "watch": "tailwindcss -i ./src/styles/tailwind.css -o ./assets/css/tailwind.css --watch"
}
```

16.4 Ejemplo de conversión de un formulario (Bootstrap → Tailwind)

Bootstrap

```
<form class="card shadow p-4">
  <input class="form-control" name="nombre" required>
  <button class="btn btn-success w-100">Enviar</button>
</form>
```

Tailwind

```
<form class="bg-white shadow rounded-lg p-6">
  <div class="mb-4">
    <label class="block text-sm font-medium mb-1">Nombre</label>
    <input name="nombre" required class="w-full border rounded-md px-3 py-2">
  </div>
  <button class="w-full bg-primary text-white py-2 rounded-md font-semibold">Enviar</button>
</form>
```

Referencias

Tailwind Labs Inc. (2025). *Tailwind CSS Documentation*. Recuperado de: https://tailwindcss.com/docs

Bootstrap Team. (2024). *Bootstrap 5 Documentation*. Recuperado de: https://getbootstrap.com/docs

The PHP Group. (2025). *PHP Manual*. Recuperado de: https://www.php.net/manual/en/

Oracle Corporation. (2025). *MySQL 8.0 Reference Manual*. Recuperado de: https://dev.mysql.com/doc/

Norman, D. (2013). *The Design of Everyday Things*. MIT Press.

Nielsen, J. (2020). *10 Usability Heuristics for User Interface Design*. Nielsen Norman Group. Recuperado de: https://www.nngroup.com/articles/ten-usability-heuristics/

Lidwell, W., Holden, K., & Butler, J. (2010). *Universal Principles of Design*. Rockport Publishers.

World Wide Web Consortium (W3C). (2025). *HTML & CSS Standards*. Recuperado de: https://www.w3.org/standards/

Mozilla Developer Network (MDN). (2025). *Web Development Documentation*. Recuperado de: https://developer.mozilla.org/

WebAIM. (2025). *Color Contrast Checker & Accessibility Guidelines*. Recuperado de: https://webaim.org/resources/contrastchecker/

Figma. (2025). *Figma Help Center*. Recuperado de: https://help.figma.com/