Memoria Técnica: Stack Tecnológico de mjr-portfolio

Generado por Asistente de IA

19 de febrero de 2026

1. Introducción

Este documento detalla las tecnologías utilizadas en el desarrollo del proyecto `mjr-portfolio`, justificando la elección de cada una en función de los requisitos de rendimiento, escalabilidad y experiencia de usuario (UX). El proyecto es un portafolio web moderno diseñado para mostrar habilidades y proyectos de manera interactiva y profesional.

2. Arquitectura Frontend

2.1. Next.js 14 (App Router)

Se ha seleccionado **Next.js 14** como el marco principal de la aplicación. Esta elección se basa en las siguientes ventajas:

- **Renderizado Híbrido:** Aprovecha el *App Router* para utilizar *React Server Components*, lo que reduce el tamaño del paquete de JavaScript enviado al cliente y mejora significativamente el rendimiento inicial de carga (Core Web Vitals).
- **SEO Optimizado:** Al renderizar el contenido en el servidor, se garantiza una indexación óptima por parte de los motores de búsqueda, crucial para un portafolio personal.
- **Enrutamiento Basado en Archivos:** Simplifica la estructura del proyecto y facilita la creación de rutas dinámicas y anidadas.

2.2. React 18

Como base de Next.js, **React 18** permite una arquitectura basada en componentes reutilizables. Esto facilita el mantenimiento del código y asegura una separación clara de responsabilidades en la interfaz de usuario.

3. Lenguaje y Calidad de Código

3.1. TypeScript

El proyecto está desarrollado íntegramente en **TypeScript**. La adopción de un sistema de tipos estático proporciona:

- **Seguridad de Tipos:** Detecta errores en tiempo de compilación, reduciendo drásticamente los errores en producción.
- **Autodocumentación:** Las interfaces y tipos definidos sirven como documentación viva del código, facilitando futuras iteraciones o colaboraciones.
- **Mejor DX (Developer Experience):** El autocompletado y la refactorización segura en el editor agilizan el desarrollo.

3.2. ESLint

Se utiliza **ESLint** con la configuración recomendada de Next.js para asegurar la consistencia del código y seguir las mejores prácticas de desarrollo.

4. Estilizado y Diseño UI/UX

4.1. Tailwind CSS

Para el estilizado se ha optado por **Tailwind CSS**. Esta librería de clases de utilidad ofrece:

- **Desarrollo Rápido:** Permite construir interfaces complejas directamente en el HTML (JSX) sin cambiar de contexto.
- **Diseño Responsivo:** Facilita la adaptación de la interfaz a diferentes tamaños de pantalla mediante prefijos simples.
- **Consistencia:** Al utilizar un sistema de diseño basado en tokens predefinidos, se asegura una coherencia visual en toda la aplicación.
- **Utilidades Adicionales:** Se emplean `clsx` y `tailwind-merge` para manejar condicionalmente las clases CSS de manera limpia y sin conflictos.

4.2. Framer Motion

Para enriquecer la experiencia de usuario, se utiliza **Framer Motion**. Esta librería permite implementar animaciones declarativas y complejas, transiciones de página y gestos, aportando una sensación de modernidad y fluidez (e.g., efectos al hacer scroll o transiciones entre secciones).

4.3. Lucide React

Se utiliza **Lucide React** para la iconografía. Esta librería proporciona un conjunto de iconos SVG consistentes, ligeros y altamente personalizables que se integran perfectamente con el sistema de componentes de React.

5. Integraciones y Servicios Externos

5.1. Formspree (@formspree/react)

Para la gestión del formulario de contacto se utiliza **Formspree**. Esto elimina la necesidad de mantener un servidor de backend dedicado solo para el envío de correos electrónicos, simplificando la infraestructura y reduciendo costos.

5.2. Cal.com (@calcom/embed-react)

La funcionalidad de agendamiento de citas se integra mediante **Cal.com**. Esta solución embebida permite a los visitantes reservar reuniones directamente desde el portafolio, sincronizándose automáticamente con el calendario del usuario.

6. Infraestructura y Despliegue

6.1. Docker

El proyecto incluye un `Dockerfile`, lo que permite contenerizar la aplicación. Esto asegura que el entorno de desarrollo sea idéntico al de producción, eliminando problemas de compatibilidad ("funciona en mi máquina") y facilitando el despliegue en cualquier proveedor de servicios en la nube que soporte contenedores.

7. Conclusión

La selección tecnológica de `mjr-portfolio` prioriza el rendimiento, la mantenibilidad y la experiencia de usuario. La combinación de Next.js y TypeScript proporciona una base sólida y escalable, mientras que Tailwind CSS y Framer Motion aseguran una interfaz atractiva y moderna. Las integraciones de terceros como Formspree y Cal.com permiten enfocar el desarrollo en el valor diferencial del portafolio sin reinventar la rueda en servicios estándar.