Forma

Descripción generada automáticamente con confianza baja

FACULTAD DE INGENIERÍA

Carrera de Ingeniería de Sistemas Computacionales

INNOVACIÓN TECNOLÓGICA EN EL DESARROLLO WEB CLINICO: MIGRACION DE WORDPRESS A UNA SOLUCION PERSONALIZADA CON REACT Y NEXT.JS

Trabajo de suficiencia profesional para optar al título profesional de:

Ingeniería de sistemas computacionales

**Autor:**

Marcos Daniel Romero Garcia

Asesor:

Mg. Fernando Alex Sierra Liñan

ORCID: 0000-0002-0687-3377

Lima - Perú

2025

# Informe de Similitud

(Copie y pegue como imagen la hoja del reporte global)

# Dedicatoria

El presente trabajo va dedicado con un gran cariño a mi padres y hermana, quienes fueron mi apoyo a lo largo de todo este camino, siempre motivándome a escalar profesionalmente año tras año. Quiero hacer una mención especial a mi madre, quien ha sido la persona que más ha valorado y fomentado mi desarrollo como profesional. Y un especial agradecimiento a mi asesor por guiarme en cada paso del desarrollo del trabajo de suficiencia profesional.

# Agradecimiento

En primer lugar, quiero agradecer a mi madre Yesy Garcia, que fue la persona que más ha fomentado en mis las ganas de seguir adelante.

A mi padre, Clever Romero, por enseñarme los valores que me inculcó desde pequeño, los cuales han sido fundamentales en los momentos más difíciles de mi vida.

Y, por último, pero no menos importante a mi hermana Daniela Romero, quien ha sido mi mayor apoyo en las buenas y en las malas a lo largo de mi vida.

**Tabla de contenidos**

[Informe de Similitud 2](#_Toc204567104)

[Dedicatoria 3](#_Toc204567105)

[Agradecimiento 4](#_Toc204567106)

[Índice de tablas 8](#_Toc204567107)

[Índice de Figuras 9](#_Toc204567108)

[Índice de ecuaciones 10](#_Toc204567109)

[RESUMEN EJECUTIVO 11](#_Toc204567110)

[CAPÍTULO I. INTRODUCCIÓN 12](#_Toc204567111)

[1.1. Contexto 12](#_Toc204567112)

[1.2. Historia de la empresa 13](#_Toc204567113)

[1.3. Datos principales de la empresa 14](#_Toc204567114)

[1.4. Localización de la empresa 15](#_Toc204567115)

[1.5. Cultura organizacional 16](#_Toc204567116)

[1.5.1. Misión 17](#_Toc204567117)

[1.5.2. Visión 17](#_Toc204567118)

[1.6. Diagrama de estructura organizacional 17](#_Toc204567119)

[CAPÍTULO II. MARCO TEÓRICO 21](#_Toc204567120)

[2.1. Modernización de Sistemas Web 21](#_Toc204567121)

[2.1.1. Evolución de las Tecnologías Web 21](#_Toc204567122)

[2.1.2 Limitaciones de los Sistemas de Gestión de Contenidos Tradicionales 22](#_Toc204567123)

[2.1.3 Principios de la Arquitectura Web Moderna 24](#_Toc204567124)

[2.1.5 Seguridad arquitecturas web modernas 27](#_Toc204567125)

[2.2. Tecnologías de Desarrollo Web Moderno 27](#_Toc204567126)

[2.2.1 React y el Desarrollo basado en Componentes 28](#_Toc204567127)

[2.2.2 Next.js como framework integral 28](#_Toc204567128)

[2.2.3 TypeScript y la robustez del código 29](#_Toc204567129)

[2.2.4 Despliegue profesional con Vercel 29](#_Toc204567130)

[2.2.5 Metodología Scrum 30](#_Toc204567131)

[2.2.6 Control de Versiones con Git y GitHub 30](#_Toc204567132)

[2.2.6 Justificación de las tecnologías utilizadas 31](#_Toc204567133)

[CAPÍTULO III. DESCRIPCIÓN DE LA EXPERIENCIA 32](#_Toc204567134)

[3.1 Proceso de selección 32](#_Toc204567135)

[3.2. Contexto del Proyecto y Personas Involucradas 32](#_Toc204567136)

[3.3. Fases de Ejecución del Proyecto 35](#_Toc204567137)

[3.3.1. Fase I: Propuesta y Planificación Estratégica 35](#_Toc204567138)

[3.3.3. Fase III: Desarrollo e Implementación por Hitos 38](#_Toc204567139)

[CAPÍTULO IV. RESULTADOS 38](#_Toc204567140)

[CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES 40](#_Toc204567141)

[REFERENCIAS 41](#_Toc204567142)

[ANEXOS 43](#_Toc204567143)

# Índice de tablas

# Índice de Figuras

[Figura 1. Reporte de la ficha RUC 14](#_Toc204569059)

[Figura 2. Mapa relieve de localización de la empresa 15](#_Toc204569060)

[Figura 3. Instalaciones del Hub Insalud 16](#_Toc204569061)

[Figura 4. Mapa satelital de localización de la empresa. 16](#_Toc204569062)

[Figura 5. Organigrama principal de gerencia 17](#_Toc204569063)

[Figura 6. Organigrama del área de investigación 18](#_Toc204569064)

[Figura 7. Organigrama del área de gestión y operaciones 1 18](#_Toc204569065)

[Figura 8. Organigrama del área de gestión y operaciones 2 19](#_Toc204569066)

[Figura 9. Organigrama del área de tecnología y transformación digital 19](#_Toc204569067)

[Figura 10. Organigrama del área de marketing y comercial 20](#_Toc204569068)

[Figura 11. Protipo de ruleta ganadora para nuevos pacientes 39](#_Toc204569069)

[Figura 12. Prototipo para la landing de VPH 40](#_Toc204569070)

# Índice de ecuaciones

(El presente índice se fijará en función a la naturaleza del trabajo. Las ecuaciones se emplean habitualmente en investigaciones en ingeniería)

# RESUMEN EJECUTIVO

Explica en 200 palabras el entorno en el cual se desarrolló la experiencia profesional precisando el proyecto o problema laboral afrontado, herramientas o modelos utilizados para el desarrollo de la solución, los resultados y las conclusiones, así como las competencias profesionales aplicadas.

# CAPÍTULO I. INTRODUCCIÓN

## Contexto

Este trabajo presenta la experiencia adquirida en la empresa Insalud Corp E.I.R.L., en la práctica de la innovación tecnológica, en específico, migración de la aplicación web clínica de Wordpress a una solución a medida construida con React.js y Next.js.

La necesidad de esta propuesta surgió a raíz de las numerosas limitaciones del anterior sistema, las cuales incluían, bajo rendimiento, mala configuración, escasa escalabilidad, dependencias de plugins y una notable falta de personalización para poder integrar funcionalidades avanzadas que vayan alineadas con los nuevos objetivos de la organización. Como desarrollador web del área, fui testigo de estas deficiencias y debido a ello es que propuse un rediseño integral basado en tecnologías más modernas y escalables ajustadas a las necesidades futuras de la empresa. Esto permitió mejorar el rendimiento, seguridad, la comunicación con el Backend y lo más importante, la experiencia del usuario.

La nueva plataforma fue diseñada e implementada utilizando herramientas como React, Next.js, GitHub, Figma, y servicios como Amazon S3 y Vercel. Actualmente, ya se encuentra en etapa de producción y sigue siendo escalada con nuevas funcionalidades y secciones. Esta iniciativa ha contribuido a fortalecer la infraestructura digital de la organización, permitiendo una administración clínica más eficiente, segura, personalizable y adaptable.

## Historia de la empresa

INSALUD es una institución de salud privada. Se fundó en el año 2020 en Lima, Perú. Ofrece servicios médicos en urología, ginecología, salud sexual y medicina regenerativa.

Desde su comienzo, la empresa ha puesto su atención en la tecnología y la prevención. Estos son pilares para dar una atención médica de calidad que se adapta a lo que los pacientes piden. Su modo de atención no solo trata enfermedades, sino que también mejora la vida de las personas. Utiliza procedimientos modernos, seguros y poco invasivos.

INSALUD ha crecido de forma constante. La gente confía en la institución. Por esto, ha podido abrir centros en otros países como Ecuador y Panamá. Su plan de negocio busca llegar a ser un modelo en salud sexual y regenerativa en América Latina. Esto se consigue al mejorar sus servicios sin parar. También, desarrolla proyectos de investigación y usa nuevas formas de trabajo.

## Datos principales de la empresa

Los datos de Insalud Corp E.I.R.L según el registro de la Superintendencia Nacional de Administración Tributaria (SUNAT) son los siguientes:

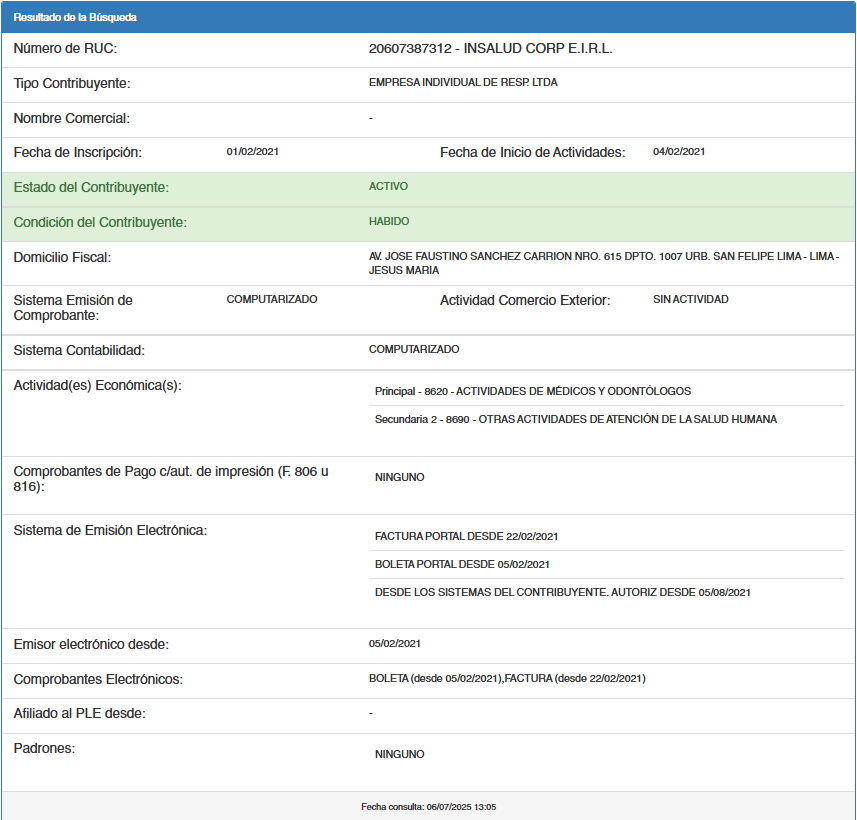


Figura 1. Reporte de la ficha RUC

*Fuente: Suma*

## Localización de la empresa

La empresa se localiza en Av. Manuel Olguín 335, Santiago de Surco 15023, en el distrito de Santiago de surco, en la provincia metropolitana de Lima, Lima, Perú. De referencia a una cuadra de la universidad de Lima.

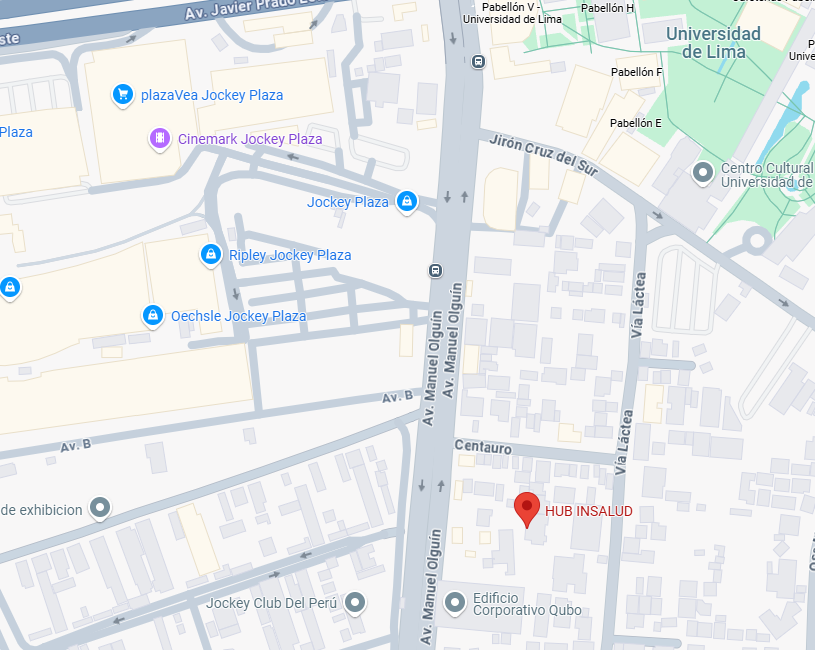


Figura 2. Mapa relieve de localización de la empresa

Fuente: Google Maps 2025



Figura 3. Instalaciones del Hub Insalud

*Fuente: Propia*

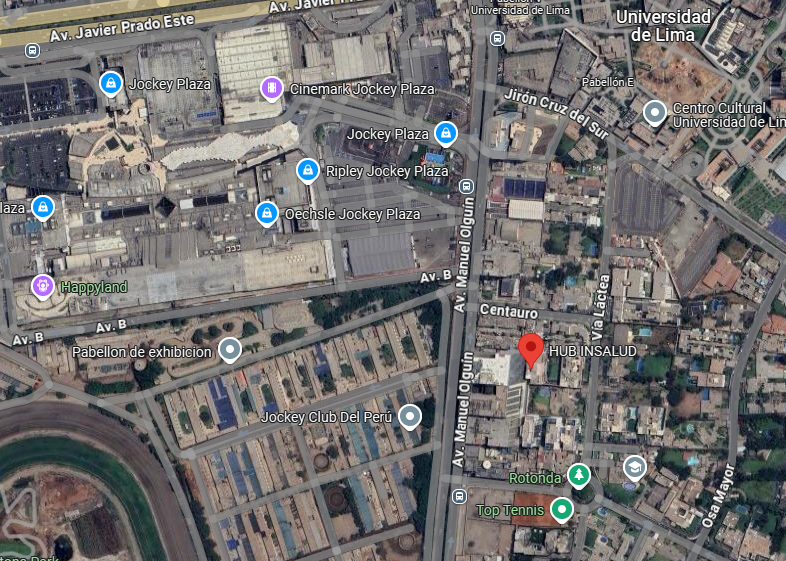


Figura 4. Mapa satelital de localización de la empresa.

Fuente: Google Maps2025

## Cultura organizacional

Insalud es una empresa dedicada a brindar atención integral en salud sexual y reproductiva, INSALUD construye su cultura organizacional sobre pilares como la empatía, la prevención, la calidad médica y el acceso informado. Se fomenta un entorno de respeto, profesionalismo e innovación constante, donde cada integrante del equipo comparte el compromiso de educar, prevenir y acompañar. Esta cultura impulsa su propósito de transformar la salud íntima en América Latina, con una visión ética, humana y cercana.

### Misión

Reducir el impacto del envejecimiento en la calidad de vida de nuestros pacientes con terapias no invasivas.

### Visión

Ser el líder regional en medicina regenerativa y antiaging, con operaciones en la mayoría de los países de LATAM para el 2027.

## Diagrama de estructura organizacional

La empresa está organizada en diferentes áreas, con el Directorio y la Gerencia General ubicados en la sede principal, junto a los departamentos de Marketing y Comercial, Administración, así como Tecnología y Cambio Digital.

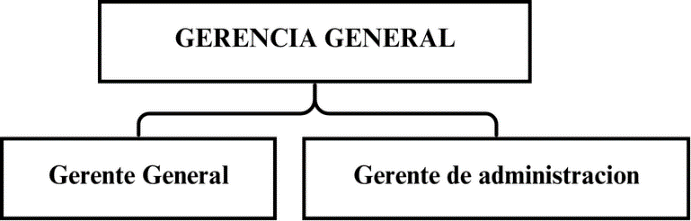


Figura 5. Organigrama principal de gerencia

*Fuente: Insalud*

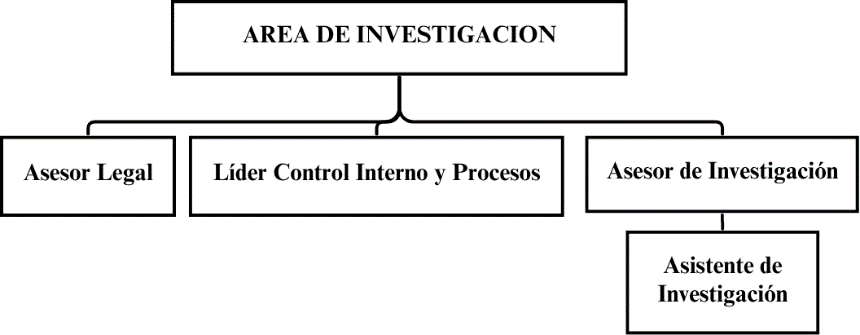


Figura 6. Organigrama del área de investigación

*Fuente: Insalud*

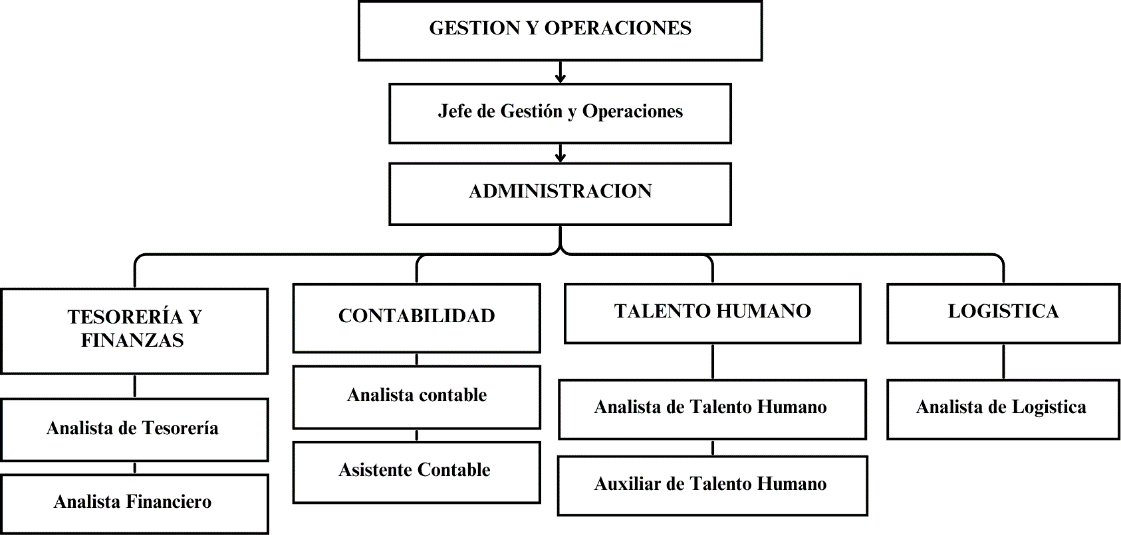


Figura 7. Organigrama del área de gestión y operaciones 1

*Fuente: Insalud*

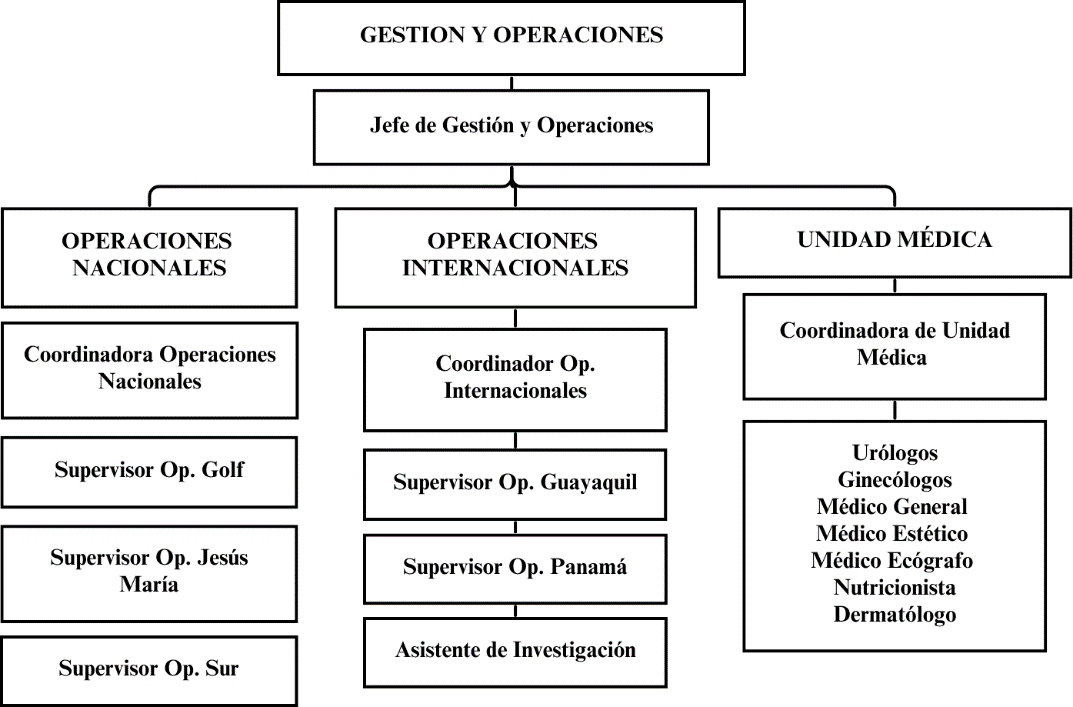


Figura 8. Organigrama del área de gestión y operaciones 2

*Fuente: Insalud*

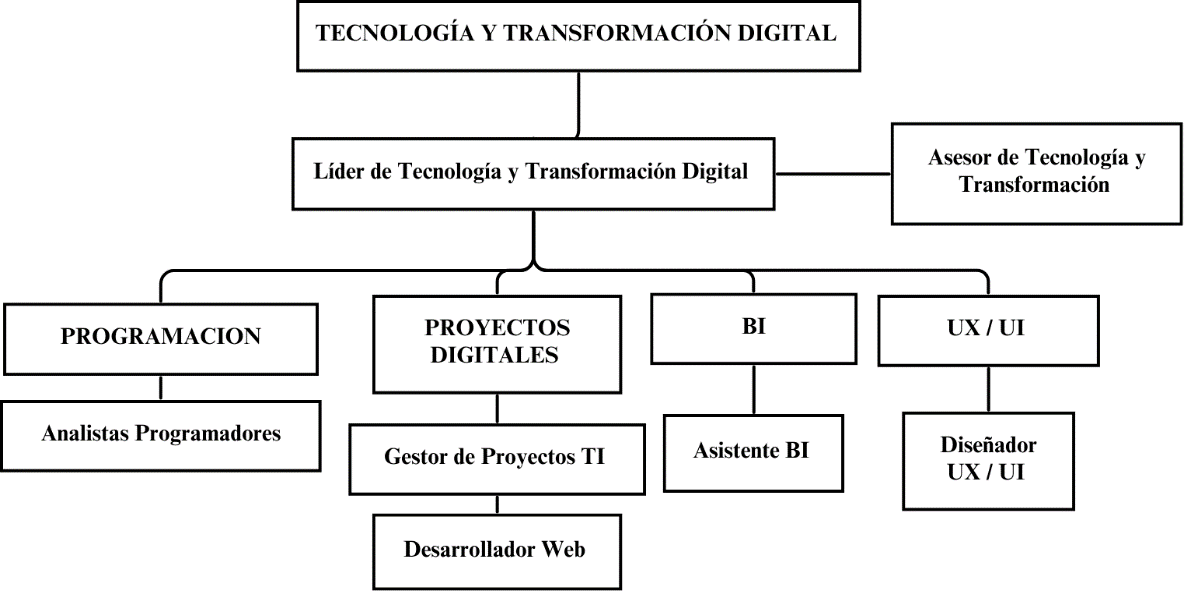


Figura 9. Organigrama del área de tecnología y transformación digital

*Fuente Insalud*

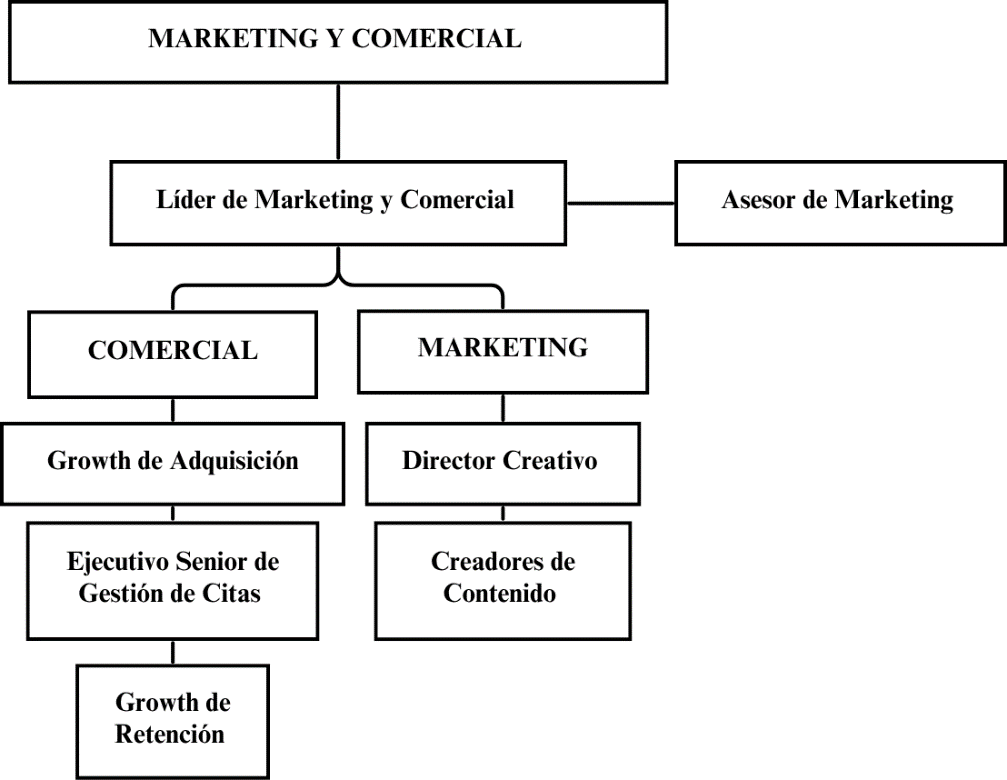


Figura 10. Organigrama del área de marketing y comercial

*Fuente: Insalud*

# CAPÍTULO II. MARCO TEÓRICO

## 2.1. Modernización de Sistemas Web

### 2.1.1. Evolución de las Tecnologías Web

El avance de las tecnologías de la web ha tenido varios cambios desde que se estableció de la World Wide Web en 1990. Según Berners-Lee et al. (2001), se estableció la web moderna, y se ofrecieron las bases esenciales que han marcado todas las orientaciones futuras en la evolución de aplicaciones web durante los últimos treinta años. El desarrollo desde las páginas web estáticos a las aplicaciones web dinámicas es un ejemplo de uno de los cambios más importante en la historia de la tecnología.

Según Garrett (2005), la evolución web se subdivide en tres generaciones: web 1.0, la cual muestra solamente contenido estático; Web 2.0 la cual adquirió interactividad con el usuario y la web misma; y web 3.0 trajo consigo tecnologías semánticas y experiencias personalizadas. Esta evolución de la web trajo consigo

Según Garrett (2005), la evolución web se puede subdividir en tres generaciones principales, a saber: web 1.0, que se define principalmente por tener contenido estático; Web 2.0 que adoptó la interactividad y la colaboración del usuario; y web 3.0 que adoptó tecnologías semánticas y experiencias más personalizadas. Esta lenta pero audaz evolución de la web ha ocasionado la necesidad de migrar desde las soluciones existentes, configuraciones y arquitecturas heredadas hasta estructuras más modernas y escalables.

La estructura web convencional, que se basa en el modelo cliente-servidor básico, ha demostrado limitaciones considerables cuando se aplica a aplicaciones empresariales complejas. Fielding (2000) introdujo el estilo arquitectónico REST. REST definía los principios clave para crear aplicaciones web escalables y mantenibles. Estos principios influyen en DreamWorks o librerías modernas como React y Next.js, que dividen la lógica del frontend y el backend y permiten relacionarse con bases de datos relacionales o no relacionales

### 2.1.2 Limitaciones de los Sistemas de Gestión de Contenidos Tradicionales

Los CMS (gestores de contenido) como lo son WordPress, Joomla, presentan múltiples limitaciones con respecto a su implementación en aplicaciones grandes, su escalabilidad no es su fuerte. Morville y Rosenfeld (2006) Se identificó que estas limitaciones derivan de la estructura sólida característica de estos sistemas, que integra la lógica de visualización, la gestión de contenidos y las funciones de la aplicación en una sola estructura cohesionada.

La dependencia de plugins y extensiones de terceros es una vulnerabilidad crítica en los sistemas CMS. Según OWASP (2021), aproximadamente el 52% de las fallas de seguridad en WordPress se originan en complementos que están mal programados o que no han sido actualizados.

Esto pone en riesgo la seguridad de las aplicaciones y al mismo tiempo genera problemas de compatibilidad, escalabilidad y rendimiento y estos van creciendo a medida que la aplicación crece. Por estos motivos, frameworks y librerías modernas como React incluyen validación segura de datos con librerías como Zod, esta ofrece verificación de tipos, validación de esquemas, etc. Esto elimina la necesidad de complementos externos para funciones esenciales.

Lo sistemas de gestión de contenido tradicionales tiene un impacto significativo debido a su renderizado del lado del servidor. Cuando un usuario solicita, es necesario que la página completa se procese, lo cual trae consigo consultas a bases de datos, renderizaciones y otra creación de HTML. Esto genera tiempos largos de espera y bajos niveles de experiencia de usuario, especialmente en aplicación webs interactivas.

Cuando se desarrolla aplicaciones web es necesario que se utilicen enfoques de gestión de fácil y flexible adaptación que garanticen entregas progresivas de características. Según Schwaber y Sutherland (2020),

El diseño de aplicaciones web actuales necesita enfoques de gestión de proyectos que permitan una flexible adaptación y una entrega progresiva de características. Según Schwaber y Sutherland (2020), La estrategia Scrum proporciona un marco ágil que es especialmente beneficioso para proyectos de desarrollo en tecnología donde los requisitos pueden modificarse durante el avance. Al aplicar Scrum en iniciativas de migración tecnológica, se permite la entrega frecuente de componentes funcionales, facilitando la validación anticipada de las decisiones arquitectónicas y la adaptación a las nuevas necesidades del negocio.

## 2.1.3 Principios de la Arquitectura Web Moderna

La manera en la que se construye la web moderna tiene como fundamentos principios como la separación de responsabilidades, escalabilidad y modularidad. Hunt y Thomas (2019) determinaron que las aplicaciones web actuales necesitan implementar diseños arquitectónicos que posibiliten el desarrollo separado de los elementos del frontend y del backend, lo que facilita su mantenimiento y escalabilidad.

El desarrollo basado en componentes es un principio fundamental de la arquitectura web moderna. Gamma et al. (1994) implementaron diseños que han tenido un impacto importante en la formación de marcos actuales como React. Esta metodología facilita la elaboración de interfaces de usuario que son modulares, reutilizables y fáciles de mantener, donde cada componente contiene su lógica particular y puede ser creado, evaluado y puesto en funcionamiento de forma autónoma.

Las Single Page Applications – SPA (arquitectura de aplicaciones de página única) representan una evolución significativa en la experiencia del usuario web. Fink y Flatow (2013) evidenciaron que las SPA ofrecen interacciones más suaves y reactivas al evitar las recargas totales de la página, conservando el estado de la aplicación en el lado del cliente y comunicándose con el servidor solo para intercambiar información concreta.

El renderizado híbrido, que combina el renderizado del lado del servidor (SSR) y del lado del cliente (CSR), ofrece beneficios optimizados tanto para el rendimiento inicial como para la experiencia del usuario posterior. Vercel (2023) ha demostrado que esta aproximación mejora significativamente los tiempos de carga inicial y la optimización para motores de búsqueda, mientras mantiene la interactividad dinámica de las aplicaciones web modernas.

**2.1.4 Beneficios de la Migración Tecnológica**

Hay variedad de mejoras operativas cuando hablamos de migración de sistemas CMS tradicionales a arquitecturas web modernas, Google (2020) registró que las aplicaciones web creadas con marcos modernos presentan incrementos promedio del 40% en los tiempos de carga inicial y del 60% en la capacidad de respuesta de la interfaz de usuario.

El aumento en el rendimiento proviene de la optimización del proceso de renderizado y de una gestión eficaz del estado de la aplicación. Las metodologías de división de código y carga diferida, que están integradas en los frameworks contemporáneos, garantizan que solo se carguen los recursos requeridos para cada vista en particular, lo que disminuye notablemente el tiempo de carga inicial y el uso de ancho de banda.

La división en módulos y la distribución de tareas aumentan significativamente la capacidad de escalar y mantener el código. Un diseño sólido promueve la identificación y solución de fallos, lo que a su vez reduce la redundancia del código y permite llevar a cabo pruebas unitarias de manera más efectiva. Estos factores contribuyen a reducir en cerca de un 30% los costos de mantenimiento a lo largo del tiempo.

La capacidad de escalar horizontalmente se logra mediante la división entre el frontend y el backend, lo que permite que cada capa se escale de forma independiente según las necesidades particulares del sistema. Esta adaptabilidad en la arquitectura es especialmente beneficiosa en contextos empresariales donde los patrones de utilización pueden cambiar notablemente entre diversos módulos de la aplicación.

Cuando creamos aplicaciones webs dinámicas y fáciles de usar, la interacción del usuario mejora considerablemente. Para garantizar experiencias continuas, similares a las de aplicaciones nativas de escritorio o incluso móviles viene cuando manejamos la

La interacción del usuario mejora con la adopción de interfaces que son más dinámicas y fáciles de usar. La habilidad de modificar partes particulares de la interfaz sin alterar el estado general de la aplicación ofrece una experiencia más continua, similar a la de aplicaciones nativas en computadoras o dispositivos móviles.

### 2.1.5 Seguridad arquitecturas web modernas

Cuando pensamos en el diseño de la arquitectura en la web moderna, la seguridad se vuelve una de las preocupaciones principales. Cuando nuestra aplicación empieza a comunicarse con APIs, el riesgo de ataques crece.

Según OWASP (2021), uno de los aspectos fundamentales para las aplicaciones actuales es la implantación de comprobaciones fuertes, autenticaciones y detalladas restricciones de acceso.

Por ese motivo, hay frameworks como Next.js, lo cuales ya cuenta con buenas y rigurosas prácticas de seguridad integradas en sí mismo, como lo son encabezados http seguridad, tokens de autenticación, jet, etcétera. Además, esto es fortalecido con el despliegue en plataformas de despliegue segura, como Vercel (OWASP, 2021; Vercel, 2024)

## 2.2. Tecnologías de Desarrollo Web Moderno

A lo largo del tiempo, la creación de aplicaciones web ha ido cambiando, teniendo ahora como tendencia un enfoque más centrado en el rendimiento, más estructurado y modular. Por ello, tecnologías como Next.js, React, TypeScript, Vercel y la adopción de metodologías como Scrum conforman los pilares para los proyectos web en la actualidad.

### 2.2.1 React y el Desarrollo basado en Componentes

React es una biblioteca basada en JavaScript, creada por Meta, esta nos permite construir interfaces de usuario mediante componentes reutilizables, además, maneja la capacidad de actualizar vistas de forma eficiente gracias a su modelo declarativo y al uso del DOM virtual. Todo en conjunto facilita el desarrollo de interfaces de manera escalable y mantenible. (Meta, 2023).

### 2.2.2 Next.js como framework integral

Next.js es un framework basado en React, desarrollado por Vercel, que ofrece un enfoque integral para construir aplicaciones web modernas. A diferencia de usar múltiples herramientas por separado, Next.js ya nos proporciona: Renderizado híbrido (SSR y SSG), Rutas basadas en archivos, API Routes para usar la lógica del servidor (Sin necesidad de una Backend adicional) Y un despliegue directo bastante intuitivo desde su propia plataforma nativa (Vercel, 2024).

Con esto hacemos más sencilla la arquitectura de nuestras aplicaciones, podemos mantener todo el desarrollo del proyecto en un solo entorno, eliminando la necesidad de herramientas que solemos usar bajo otros enfoques, tales como Vite, Web pack o inclusive bases datos externas si no se requieren,

### 2.2.3 TypeScript y la robustez del código

TypeScript extiende las capacidades actuales del lenguaje de programación JavaScript brindándonos algo de lo que JavaScript carece lo cual es un tipado estático. Esto añade una capa de seguridad si hablamos de errores comunes en el desarrollo, nos mejora el autocompletado en editores de código y como ya mencionamos, al tener un tipado más estricto, nuestro proyecto crece con mucha más robustez (Microsoft, 2024). En combinación con Herramientas anteriormente mencionadas, como React y Next.js, TypeScript, nos permite tener un desarrollo mucho más mantenible

### 2.2.4 Despliegue profesional con Vercel

Vercel, es la plataforma oficial para poder desplegar proyectos desarrollados en Next.js. Nos ofrece muchas ventajas, entre ellas tenemos, Integración directa con GitHub, CI/CD automatizado, actualizaciones inmediatas a través de git mediante “push”, y un muy intuitivo manejo de dominios personalizados. También posee su modelo serveless, esto elimina la necesidad de configurar servidores, con esto reducimos el tiempo de puesta en producción (Vercel, 2024).

### 2.2.5 Metodología Scrum

Sin ninguna duda, Scrum una de las metodologías agiles más esparcidas a nivel mundial y por ende al mismo tiempo se convierte en una de las más utilizadas en el desarrollo de software. Scrum organiza todo el trabajo en Sprints e involucra roles, tales como: Scrum Master, Product Owner y equipo de desarrollo), también da lugar a reuniones claves como: dailys, stand-ups, sprint planning, reviews y retrospectives. Esta estructura de trabajo contribuye a la mejora, transparencia y capacidad de adaptación a cambios en los requisitos (Schwaber & Sutherland, 2020).

### 2.2.6 Control de Versiones con Git y GitHub

En el desarrollo profesional de aplicaciones el uso de sistemas de control de versiones se ha vuelto algo sencillamente esencial. Git, es un sistema de control de versiones que permite llevar un historial de los cambios realizados a nivel de código en el proyecto, esto nos facilita la colaboración, ya que, nos permite trabajar con ramas paralelas y resolver conflictos (Chacon & Straub, 2014). GitHub, por su lado, nos brinda una plataforma donde nuestros proyectos pueden ser guardados, además de poder gestionarlos y compartidos también nos ofrece herramientas para poder revisar nuestro código, haces seguimiento a las issues y automatización de flujos.

En proyectos como este, GitHub se vuelve esencial, ya que más allá de simplemente almacenar nuestro código nos permite combinar los flujos de integración y despliegue continuo con Vercel, lo cual acelera los tiempos de entrega y minimiza los errores humanos.

### 2.2.6 Justificación de las tecnologías utilizadas

Combinar Next.js, React, TypeScript, así como el despliegue en Vercel y manejar una metodología ágil como Scrum, además de seguir buenas prácticas en el desarrollo web profesional, permite entregar productos escalables, optimizados, veloces, sin necesidad de aplicar configuraciones complejas ni depender de demasiadas tecnologías.

# CAPÍTULO III. DESCRIPCIÓN DE LA EXPERIENCIA

Una vez culminadas mis prácticas profesionales en BanBif, me dispuse a buscar retos más grandes, no solo para adquirir experiencia como lo había hecho previamente en la entidad bancaria, mi objetivo era proponer soluciones y mejoras en los procesos, siempre alineado con la rama en la que me estoy constantemente especializando: el desarrollo de Software.

## 3.1 Proceso de selección

Algunos días antes de mi salida del banco, ya me encontraba yo explorando en las diversas plataformas de empleo tales como LinkedIn, Getonboard, Bumeran, entre otras. Al final, encontré una oportunidad con la empresa Insalud Corp E.I.R.L la cual confió en mi para el puesto de Desarrollador Web. Luego de una agradable charla con Valeria Sanches (Analista de atracción de talento) y la entrevista con el equipo comercial y ti, pude dar inicio a mis funciones un 14 de abril del año 2025.

### 3.2. Objetivos del Proyecto

#### 3.2.1. Objetivo General

Migrar todo el sitio web de la clínica insalud de un sistema de contenido en Wordpress a una solución moderna y escalable con React y Next.js, con el fin de mejorar el rendimiento, la escalabilidad y la recolección de leads.

#### 3.2.2 Objetivos Específicos

1. Diagnosticar limitaciones técnicas y de rendimiento en la web existente alojada en WordPress para justificar la necesidad de la migración a código de React y Next.js.

2. Diseñar y desarrollar un producto mínimo viable para tres landings internacionales (Ecuador, Perú, Panamá) con React.js y Next.js.

3. Implementar lógica en el backend para procesar formularios de contacto con Next.js y NodeMailer,

### 3.3. Contexto del Proyecto y Personas Involucradas

La empresa estaba en pleno crecimiento, ya se contaba con un sistema interno desarrollado y óptimo para la gestión y recolección de los datos de los pacientes atendidos. Todo esto fue desarrollado por un analista programador y se le daba revisión y mantenimiento a lo largo de todas las jornadas, así mismo, se contaba con un analista de datos, el cual era el encargado de tomar los datos de todos los pacientes para fines de análisis.

#### 3.2.1. Diagnóstico Inicial y Justificación del Proyecto

A pesar de tener un sistema interno para la gestión de pacientes, no se contaba con una web completamente funcional para la captación de pacientes, la página que se tenía en ese momento estaba construida en WordPress y se tenía mucha información desactualizada, tales como ofertas de trabajo vencidas, algunos doctores que ya no trabajaban para la empresa, nuevas especialidades no publicadas, sedes no agregadas, etc.

Estuve analizando el flujo, apariencia y performance de esta web y me encontré con múltiples problemas, en primer lugar, las métricas de Google no eran favorables en aspectos como seo, accesibilidad y rendimiento. Esto tiene como consecuencias que los posibles clientes no lleguen de manera orgánica, que cierto público con necesidades especiales no tenga la mejor experiencia y un bajo performance al momento de visualizar la web en el teléfono móvil.

Por el lado técnico, la web construida en WordPress carecía de personas que la administren, no tenía contenido nuevo desde que se creó, la gestión de plugins se había vuelto innecesariamente tediosa por la numerosa cantidad, habían formularios vinculados a correos a los que nadie tenía seguimiento, la personalización del sitio en lo que se refiere a estilos se veía bastante limitada debido a la poca flexibilidad del plan gratuito del plugin de Elementor, no tenía conexión al sistema interno de la clínica, se carecía de un versionamiento que agilizara la revisión de cambios, se encontraba completamente aislado de los datos obtenidos del sistemas interno y una nula innovación antes de mi llegada.

En base a todo lo descrito planteé la idea de una reestructuración total de toda la web Insalud existente hasta la fecha de la revisión, utilizando tecnologías modernas y escalables que solucionen los problemas actuales y sea más flexible a cambios en el futuro.

#### 3.2.2. Funciones Desempeñadas y Equipo del Proyecto

Para poder iniciar con el proyecto de migración, el equipo estuvo conformador por:

Sebastián Palmieri, Growth Hacking: Responsable de analizar el comportamiento del usuario y dar el alcance de las métricas del negocio para definir los requerimientos funcionales, fue el responsable principal de brindar las características necesarias del proyecto.

Stivo Valladares, UI/UX: Responsable de transformar todos los requerimientos y estrategias del Growth Hacking en diseños visuales en Figma, asegurando así una interfaz atractiva, intuitiva y accesible por los futuros posibles clientes.

Marcos Romero, Desarrollador Web: como único responsable técnico, abarque todo el ciclo de vida del desarrollo de la aplicación a partir de los requerimientos y diseño planteados por el equipo de marketing, esto incluyó:

Arquitectura de la solución: Liderar la propuesta y justificación del nuevo proyecto desarrollado en React y Next.js

Desarrollo frontend y backend: En base a todos los diseños en Figma, construir componentes reutilizables y escalables en React y Next.js. Además, construir un backend ligero para el envió de correos con NodeMailer.

Optimizacion y gestión de despliegue: Asegurar que la web cuente con los más altos estándares de rendimiento (Core Web Vitals), accesibilidad (normas WCAG) y optimizado para motores de búsqueda (SEO On-page). Además, administrar el versionamiento con Git y manejar el despliegue continuo en Vercel.

## 3.3. Fases de Ejecución del Proyecto

El proyecto se llevó a cabo con un fin estratégico, el cual era modernizar la presencia de Insalud en los medios digitales. A fin de explicar las fases lógicas que representan el ciclo de vida del proyecto, el informe ha sido estructurado en cuatro fases lógicas, 1) Propuesta y planificación estratégica, 2) Diseño de la solución, 3) Desarrollo e implementación por hitos, 4) Pruebas y despliegue.

### 3.3.1. Fase I: Propuesta y Planificación Estratégica

**El Diagnóstico como Punto de Partida**

Tras un exhaustivo análisis de la web anterior creada en WordPress, pude identificar problemas críticos, tales como, bajo performance, escasa personalización, ningún tipo de integración con el backend y carencia de actualizaciones.

Como ingeniero en sistemas sentí la necesidad de no solamente aminorar los efectos negativos de la situación, sino también, reestructuras todo el proyecto de sus bases para que en el futuro cualquier tipo de interacción sea mucho más interactiva tanto desde el equipo técnico, así como de los futuros clientes.

Por ello, convoque una reunión con el equipo de Marketing, principalmente con el Growth Hacking para para presentar los hallazgos y plantear la idea de una migración total de toda la web a una solución en React y Next.js. La respuesta fue completamente positiva, al equipo le encanto la idea, de paso se añadió que la nueva web que se iba a construir iba a tener un seguimiento bastante exhaustivo con Google Tag Manager y Google Analitycs 4 para analizar el nivel de aceptación y los leads que se iban a obtener después de esta migración.

**Definición de Objetivos y Alcance del MVP**

Aunque la visión a largo plazo es migrar todo el sitio web, lo cual es lo más optimo, se optó por abordar una estrategia MVP (Producto mínimo viable) para demostrar valor al proyecto de una manera mucho más ágil. La clínica gana en base a la cantidad de pacientes que se atienden en ella, por ello, el objetivo principal era la generación de leads a través de WhatsApp y formulario de contacto. Este alcance se definió en la creación de tres landings pages internacionales, específicamente enfocados a las sedes de Perú, Panamá y Ecuador, cada una de ellas iba a tener un enfoque distinto, ya que cada una iba a orientada a un padecimiento específico.

1. **Perú:** Virus del Papiloma Humano (VPH).
2. **Panamá:** Disfunción Eréctil.
3. **Ecuador:** Prostatitis.

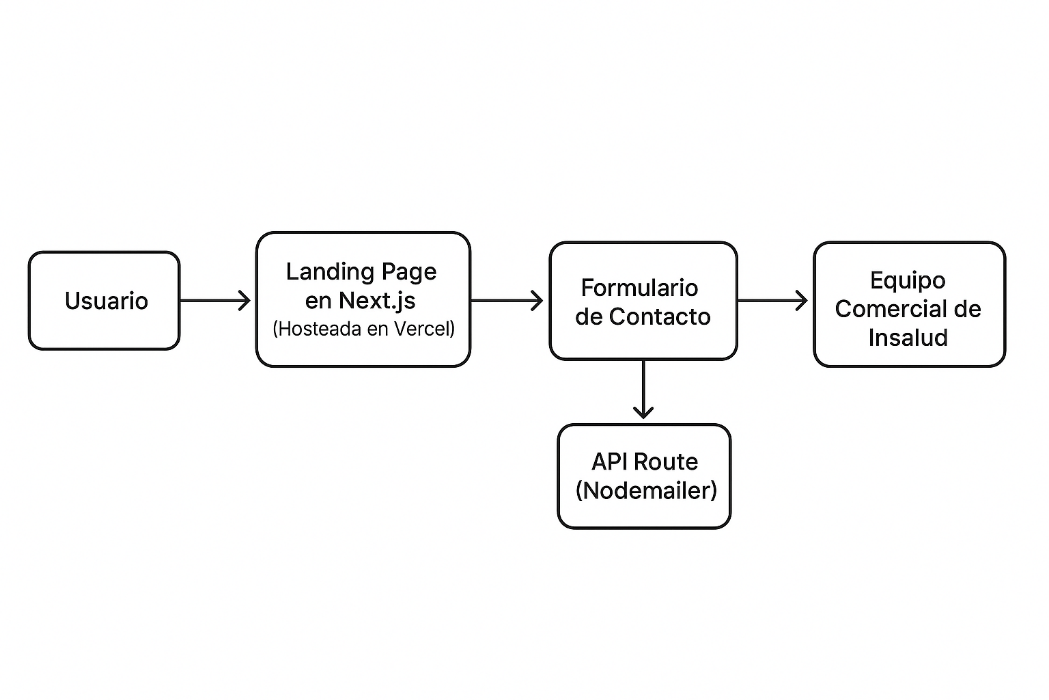
**Adquisición de Herramientas Clave**

Para poder viabilizar el proyecto gestiones junto con el UI/UX la adquisición de licencias y servicios necesarios para el desarrollo del producto, los cuales fueron aprobados por el área.

**Servicio Amazon S3:** Ya se contaba con este servicio, sin embargo, fue necesario solicitar un acceso al equipo TI. Este servicio nos proporciona un almacenamiento en la nube diseñado para servir Imágenes y PDF (Entre otros archivos) en nuestras landings.

**Figma:** Para poder permitir y tener un diseño de UI/UX profesional y colaborativo

**Vercel:** Nos sirve como plataforma de despliegue por su integración nativa con Next.js, facilidad de uso y capacidad de despliegue continuo a través de Git y GitHub.



### 3.3.2. Fase III: Desarrollo e Implementación por Hitos

**Proceso Colaborativo de Diseño**

En esta etapa, el diseño de las interfaces fue llevado a cabo con una dinámica colaborativa. Sebastián, Growth Hacking aporto de su lado los insights y los objetivos de conversión para cada landing. Con esta información, Stivo, diseñador UI/UX, elaboro los mockups y prototipos en figma, priorizando claridad, jerarquía visual y experiencia del usuario.

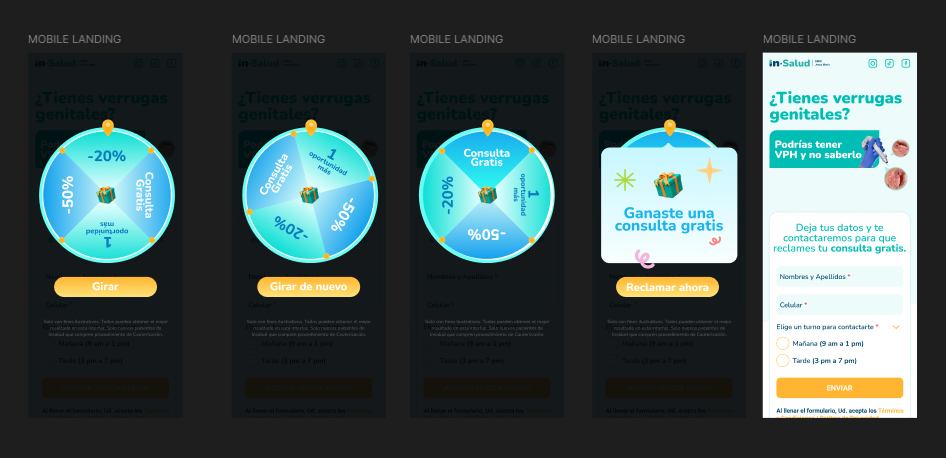


Figura 11. Protipo de ruleta ganadora para nuevos pacientes

*Fuente: Figma*

Durante toda esta fase estuve participando de manera activa, asegurándome de que cada propuesta de diseño sea viable en desarrollo con el stack que se contaba (React, Next.js), alineándolas a tiempos de carga bajos y optimo rendimiento. De hecho, en el inicio se tenía planteado una gran variedad de efectos parallax y animaciones con el scrooll para las landings, sin embargo, en conjunto con el equipo, dejamos estas animaciones para el futuro, la intención era salir con el MVP en un tiempo optimo, por esto solo se optó por animaciones básicas. Además, incluí sugerencias para optimizar componentes, estructurar mejor los formularios y solucionar cualquier problema de implementación que se pueda presentar.

Este trabajo fue un equilibrio entre la creatividad, conversión y viabilidad, elementos fundamentales para que la implementación fuese llevada a cabo ágilmente en la siguiente fase del proyecto.

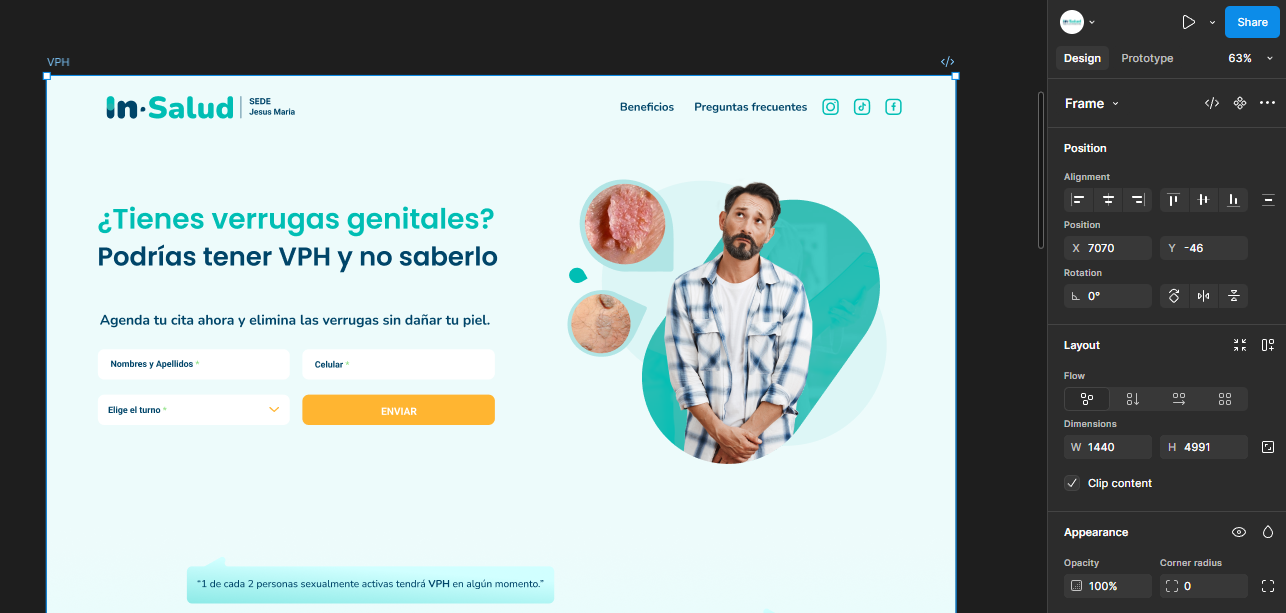


Figura 12. Prototipo para la landing de VPH

*Fuente: Figma*

### 3.3.3. Fase III: Desarrollo e Implementación por Hitos

**Metodología de Desarrollo y Gestión de Tareas**

Para organizar el desarrollo de las landings internacionales (Perú, Ecuador y Panamá), se llevó la gestión de tareas en trello. Se aplico un marco ágil inspirado en la ya conocida metodología scrum. A pesar de tener un equipo reducido, se siguió con esta estructura por cuestiones de disciplina y agilidad.

Cada una de las landings se trató como una épica y sus componentes fueron desglosados en tareas individuales (Header, Formulario, Sección de Beneficios, etc.) El tablero en trello fue organizado en columnas que permitían mostrar el flujo de un sprint: Product Backlog (repositorio de ideas y requisitos), Sprint Backlog (tareas para el ciclo actual), En Desarrollo, En Revisión (QA)) y Hecho (Done).

A pesar de ser el único desarrollador en el proyecto, se utilizó un trabajo basado en ramas con Git (develop y feature/nombre-tarea), esto me permitía a mi y al equipo tener un sistema de control de versiones mucho más organizado, así, nos facilitaba la integración y el despliegue continuo.

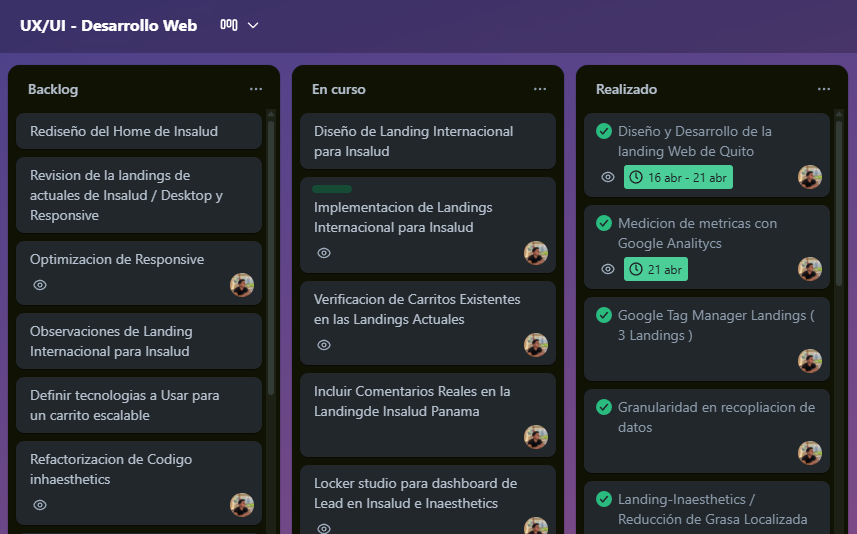
**

Figura 13. Ejemplo de la gestión de tareas en Trello para el desarrollo de las landings pages.

Fuente: Trello

**Hitos de Implementación Técnica**

Todo el desarrollo técnico se basó en tres logros claves:

Componentes reutilizables: Viendo la gran cantidad de elementos creados por el UX/UI, se creó una librería interna que incluía los elementos atómicos que la conformaban, tales como, botones, cta, inputs y cards, así como componentes mas complejos como lo es el layout principal y el formulario.

Lógica del servidos: Con la ayuda de Next.js se implementó una API route para que lo formularios puedan ser procesados eficientemente. La ruta del lado del Backend recibía los datos del cliente (Nombres y Teléfono), validaba los datos y utilizaba nodemailer para enviar un correo a modo de notificación de un nuevo lead al equipo de gestores encargados de atender las reservas. Esta fue una solución de bajo costo y buen rendimiento.

Integración de Analítica Web: Mas allá de ser un requerimiento solicitado por el equipo de Growth, fue para mí un reto personal y al mismo tiempo una propuesta de valor clave para el proyecto. Sucede que era la primera vez que yo implementaba un seguimiento de conversiones a nivel de eventos, se definieron nomenclaturas y se inició la medición desde 0. Google Tag Manager se implemento en cada una de las landings para realizar un seguimiento detallado. Esto permitió al equipo Growth tener una medición precisa del ROI de las campañas de Marketing.

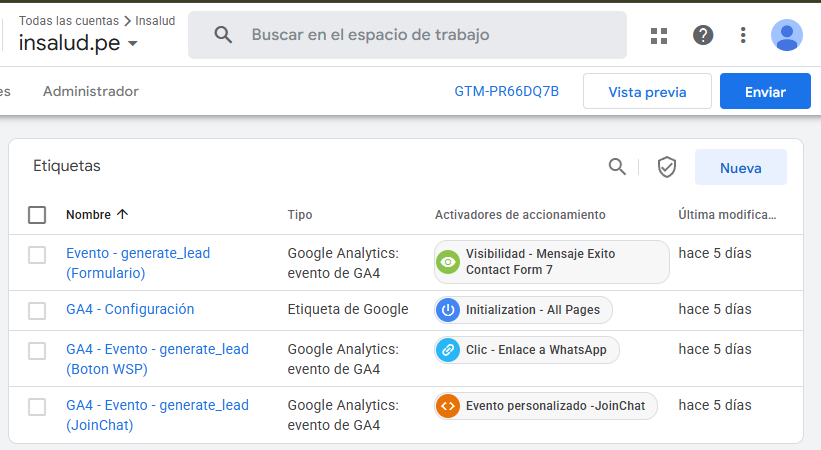
**

Figura 14. Etiquetas implementadas en Google Tag Manager para el seguimiento de leads

*Fuente: Google Tag Manager*

### 3.3.4. Fase IV: Pruebas y Despliegue

**Despliegue para Pruebas y Feedback**

Gracia a la integración que tiene Vercel con GitHub, cada cambio a una rama en Git generaba automáticamente un enlace para visualizar el avance de la página, esto permitía obtener Feedback inmediato por parte de Sebastián (Growth Hacking) y Stivo (UI/UX). Esta iteración fue clave, por ejemplo, tras una revisión de los formularios con el Growth Hacking se nos sugirió a mi y a el UI/UX reducir los campos del formulario de contacto, el cual contaban con mas campos tales como, fecha, turno, correo. Esto se hizo para acortar el tiempo de completado de el formulario por el cliente y aumentar la tasa de conversión.

**Lanzamiento en Subdominios**

Una vez que la landings pages fueron aprobadas se le asigno un subdominio a cada una de ellas: app.insalud.pe, app.insalud.ec y app.insalud.pa. Se opto por esto en lugar de reemplazar directamente el dominio principal, esto se hizo con la intención de no interferir en el funcionamiento de la web actual en WordPress que seguía operativa.

El lanzamiento fue exitoso, durante el segundo trimestre del año (Q2), posterior al lanzamiento de las nuevas páginas, se registraron 267,987 visitantes que generaron 6,081 leads, alcanzando una tasa de conversión del 2.27%. Esta cifra representa una mejora sustancial en comparación con el primer trimestre (Q1), donde se obtuvieron 1,629 leads con una tasa de conversión del 1.68%, demostrando la efectividad de la nueva solución para capturar y convertir clientes potenciales.

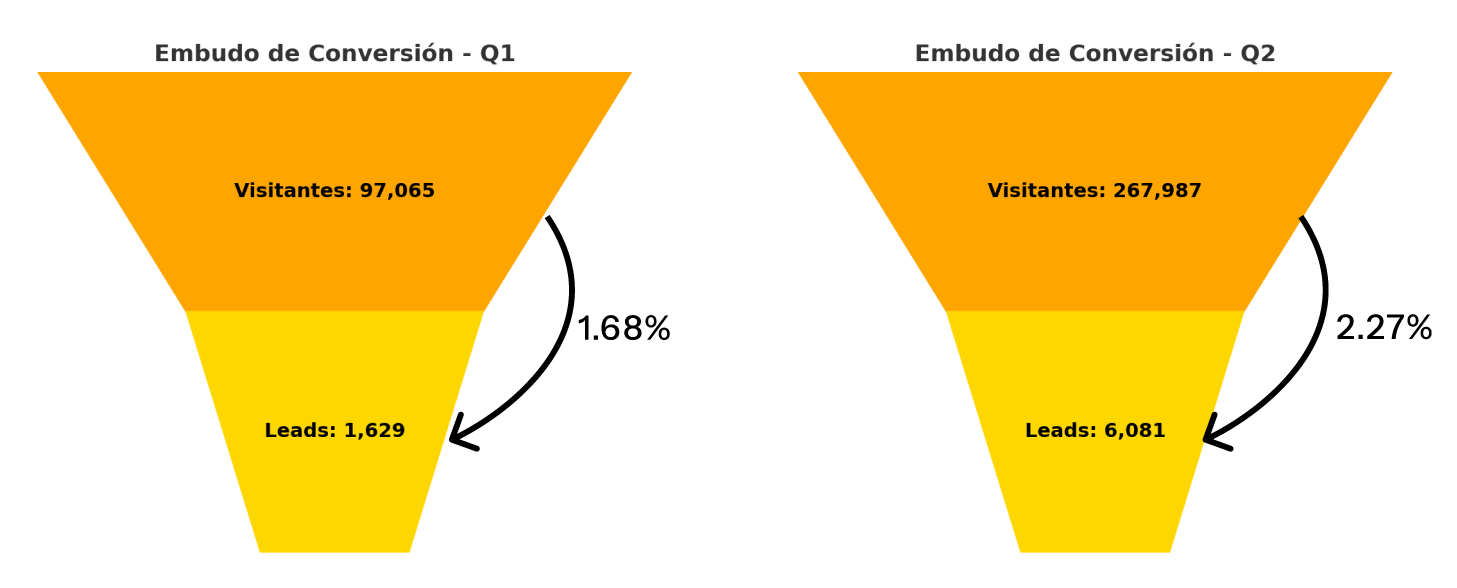


Figura 15. Embudos de conversión Q2

*Fuente: Figma*

### 3.5 Consideraciones Éticas

### 

# CAPÍTULO IV. RESULTADOS

Presenta y analiza los resultados obtenidos con el proyecto laboral. Se puede hacer uso de tablas y figuras (en algunos casos ecuaciones), sin redundar en la información presentada.

Usar el formato de tabla según estilo APA 7.

Ejemplo:

Tabla

Descripción generada automáticamente

Usar el formato de figura según estilo APA 7.

Ejemplo:

Imagen que contiene Gráfico

Descripción generada automáticamente

***Nota****: para esta sección se recomienda considerar un máximo de 5 páginas.*

# CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

**Conclusiones**

Presentar las conclusiones del proyecto desarrollado en la experiencia profesional y las lecciones aprendidas. Las conclusiones deben sustentar cuáles y cómo aplicó sus competencias profesionales en su experiencia laboral.

**Recomendaciones**

Presenta recomendaciones en base a las lecciones aprendidas

# REFERENCIAS

Berners-Lee, T., Handler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, 284(5), 34–43. <https://www.scientificamerican.com/article/the-semantic-web/>

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress. <https://git-scm.com/book/en/v2>

Facebook. (2015). *Introduction to GraphQL*. https://graphql.org/learn/

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures* (Doctoral dissertation, University of California, Irvine). <https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>

Fink, C., & Flatow, M. (2013). *Single Page Web Applications: JavaScript end-to-end*. Manning Publications. https://www.manning.com/books/single-page-web-applications

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Garrett, J. J. (2005). *Ajax: A new approach to web applications*. Adaptive Path. <https://adaptivepath.org/ideas/ajax-new-approach-web-applications/>

Google. (2020). *Web Vitals*. https://web.dev/vitals/

Hunt, A., & Thomas, D. (2019). *The Pragmatic Programmer: Your Journey to Mastery* (20th Anniversary ed.). Addison-Wesley.

Meta. (2023). *React – A JavaScript library for building user interfaces*. <https://react.dev/>

Microsoft. (2024). *TypeScript: JavaScript with syntax for types*. <https://www.typescriptlang.org/>

Morville, P., & Rosenfeld, L. (2006). *Information Architecture for the World Wide Web* (3rd ed.). O'Reilly Media.

OWASP. (2021). *OWASP Top Ten Web Application Security Risks 2021*. https://owasp.org/Top10

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. <https://scrumguides.org/>

Vercel. (2023). *Next.js Documentation*. https://nextjs.org/docs

Vercel. (2024). *Deploy web applications with Vercel*. https://vercel.com/docs

Sobre los estilos de redacción:

El formato de la tesis, las citas y las referencias se harán de acuerdo con el Manual de Publicaciones de la American Psychological Association séptima edición, los cuales se encuentran disponibles en todos los Centros de Información de UPN, bajo la siguiente referencia:

**Código**: 808.06615 APA/D

También pueden consultar la siguiente página web:

<http://www.apastyle.org/learn/tutorials/index.aspx>

Nota importante: La Facultad de Salud (excepto Psicología) utilizará el estilo Vancouver.

# ANEXOS

En los anexos colocar sólo aquellos complementos que signifique evidencias o procedimientos realizados en la investigación de manera pertinente y suficiente. (Evitar excesos innecesarios)

Cada evidencia en los anexos va en hoja independiente.

Cada hoja que contenga un anexo debe ser numerada: ANEXO N° 1. Título del anexo.