Proyecto de Grado: El Taita Jajoy - Un Puente Digital a la Medicina Ancestral

Autor del Proyecto:

Juan David Montealegre Guzmán

Universidad Libre, Sede Bosque

Ingeniería de Sistemas

Ingeniería de Software II

Ing. Rodrigo Castro Caicedo

Semestre: Cuarto

Bogotá D.C.

2025

Contenido

I. Introducción y Contexto

El proyecto **"Consultorio del Taita Jajoy"** es mi propuesta de grado enfocada en la **transformación digital cultural**. Estoy aplicando los conocimientos de Ingeniería de

Software para construir una plataforma web que modernice la gestión de consultas, productos y servicios del médico tradicional indígena Taita Jajoy. Esto no solo optimiza sus procesos operativos, sino que también es un acto de **dignificación ancestral**, llevando el saber tradicional a una audiencia digital de manera eficiente.

Esta iniciativa está dirigida a ser una solución tecnológica de bajo costo para una **microempresa** (el consultorio del Taita), enfocándome en la arquitectura **JAMstack + Serverless** para asegurar un rendimiento de alto nivel a un costo operativo mínimo, cumpliendo con la exigente línea de tiempo del semestre (Agosto a Noviembre 2025).

El proyecto se encuentra desplegado en su fase de desarrollo en: *https://consultorio-taita-jajoy.netlify.app*.

II. Definición y Alcance

II.A. Objetivo General del Proyecto

Diseñar e implementar una aplicación web **funcional, escalable y segura** que centralice la gestión de citas, pacientes y contenidos. La plataforma integra mecanismos de comunicación automatizados como:

- Notificaciones de alta fiabilidad por EmailJS.
- Comunicación directa y rápida vía redirección a WhatsApp.
- Autenticación segura mediante Google OAuth (RF01).

II.B. Lo que Resuelve el Proyecto

El sistema busca resolver dos grandes desafíos para esta microempresa:

1. **Ineficiencia Operativa:** El Taita Jajoy gestiona sus actividades de forma manual. El proyecto automatiza el agendamiento (RF07) y las notificaciones (RF19), eliminando errores humanos, optimizando su tiempo y liberándolo para la atención directa.
2. **Brecha de Acceso Digital:** Brinda a los pacientes un canal moderno y seguro para acceder a los servicios de medicina ancestral, rompiendo barreras geográficas y de comunicación.

II.C. Personas a las que está Dirigida

El sistema está diseñado para:

Grupo Objetivo	Necesidad Resuelta por el Sistema	Funcionalidades Clave
Taita Jajoy (Administrador)	Necesita control total sobre su agenda, eliminación de errores humanos y registro de información clínica sensible.	Panel administrativo para gestión de citas (RF13), horarios, y registro de diagnóstico (RF16).
Pacientes (Usuarios Finales)	Buscan un proceso simple para agendar 24/7 y recibir recordatorios confiables.	Agendamiento autónomo (RF07), autenticación segura (RF01), notificaciones automáticas (RF19) y asistente virtual basado en IA.

III. Casos de Uso Clave y Requerimientos

III.A. Casos de Uso Clave

Estos son los flujos principales que la aplicación soporta:

- **Agendamiento Autónomo 24/7:** El paciente selecciona un servicio, consulta la disponibilidad en tiempo real (RF08) a través de un calendario interactivo y confirma la reserva.
- **Soporte Inteligente (IA):** El paciente interactúa con un Asistente Virtual basado en Google GenAI (Gemini) para resolver dudas iniciales sobre los servicios o la medicina ancestral.
- **Comunicaciones de Alta Fiabilidad:** Tras agendar, el paciente recibe confirmación por correo y un recordatorio por WhatsApp. El sistema debe garantizar una entrega exitosa del 95% (RNF14).
- **Gestión Administrativa del Taita:** El Taita accede a un panel seguro para visualizar su agenda diaria, modificar horarios y registrar el diagnóstico o las recomendaciones por paciente (RF13, RF16).

III.B. Requerimientos del Sistema

Tipo	**ID**	**Descripción del Requerimiento**
Funcionales	RF01	Registro y Autenticación de Usuarios (Correo o Google OAuth).
	RF07	Sistema de Reservas: Selección de fecha y hora mediante calendario interactivo.

Tipo	ID	Descripción del Requerimiento
	RF16	Registro de Diagnóstico: Campo de texto para notas clínicas y recomendaciones.
	RF19	Automatización de Notificaciones: Envío automático por correo y WhatsApp.
No Funcionales (RNF)	RNF05	Escalabilidad: Soporte para múltiples usuarios concurrentes sin pérdida de rendimiento.
	RNF11	Rendimiento: Carga de agenda en menos de 3 segundos.
	RNF15	Consistencia: Los cambios en la disponibilidad deben reflejarse en menos de 5 segundos.

IV. Arquitectura y Diseño

IV.A. Tipo de Diseño: Patrones de Comportamiento y UX Sanitaria

El sistema propuesto se fundamenta en patrones de diseño de tipo **comportamiento** (behavioral), según la clasificación de Refactoring.guru (2024), ya que se enfoca principalmente en la forma en que los usuarios interactúan con el sistema y en cómo se distribuyen las responsabilidades entre los componentes del frontend.

Se han identificado los siguientes patrones aplicados:

- **Strategy**: La interfaz se adapta dinámicamente según el tipo de usuario (paciente o administrador de la microempresa), personalizando las funcionalidades visibles.

- **Command**: Las acciones ejecutables dentro del sistema (como agendar, cancelar o reprogramar una cita) se modelan como comandos separados, promoviendo una arquitectura limpia y extensible.
- **Observer** *(opcional)*: Si se implementan notificaciones o actualizaciones en tiempo real, este patrón facilitaría la sincronización entre componentes y la respuesta reactiva ante cambios del sistema.

Complementariamente, se adopta un enfoque de **diseño centrado en el usuario**, específico del ámbito de la salud, conocido como **UX Sanitaria**. Este enfoque busca reducir la carga cognitiva de los usuarios, generar confianza y facilitar la adopción de la herramienta por parte de personas con distintos niveles de alfabetización digital, en especial pacientes y personal de salud en microempresas.

Aunque el diseño UX Sanitario no forma parte del catálogo tradicional de patrones de diseño (GoF), puede considerarse un **idiom** o buena práctica adaptada al dominio clínico y validada por su impacto en la experiencia de usuario.

IV.B. Tipo de Arquitectura: JAMstack, Serverless y Arquitectura Basada en Eventos

La arquitectura del sistema responde a varios patrones arquitectónicos reconocidos, alineándose con principios de **modularidad, escalabilidad y bajo costo operativo**, fundamentales en entornos de microempresa.

Arquitectura JAMstack + Serverless

La solución se basa en el paradigma **JAMstack** (JavaScript, APIs, Markup) y funciones **Serverless**, lo que implica una separación clara entre el frontend y la lógica de negocio. Cada función del backend se ejecuta bajo demanda, optimizando el consumo de recursos.

Clasificación según patrones arquitectónicos

- **Arquitectura basada en eventos (Event-Driven Architecture)**: Las funciones del backend se activan ante eventos específicos, como solicitudes de agendamiento, confirmación o cancelación. Este enfoque garantiza elasticidad y respuesta inmediata ante picos de carga (por ejemplo, durante campañas de salud o temporadas altas).
- **Arquitectura orientada a servicios (SOA)**: Las funciones Serverless están diseñadas bajo principios de responsabilidad única, actuando como servicios independientes y desacoplados que pueden evolucionar o escalar individualmente.
- **Arquitectura en capas (Layered Architecture)**: El sistema presenta una separación lógica y técnica entre sus componentes:
    - **Capa de presentación**: Implementada con React y Vite, ofrece una interfaz moderna y responsiva.

- **Capa de negocio**: Compuesta por funciones Serverless, que procesan la lógica principal del sistema.
- **Capa de persistencia**: Basada en PostgreSQL, una base de datos transaccional robusta que asegura consistencia (ACID).

Esta arquitectura soporta la escalabilidad automática (RNF05), reduce costos operativos al eliminar la necesidad de servidores dedicados, y mejora el mantenimiento y despliegue al utilizar componentes desacoplados.

V. Gestión Profesional: Tiempos y Costos

El proyecto está siendo desarrollado bajo un cronograma ajustado de 4 meses, que demuestra la aplicación de metodologías ágiles para entregar un MVP funcional en un tiempo limitado.

V.A. Cronograma de Desarrollo

Fase	**Duración**	**Hitos Clave**
Fase I: Diseño y Arquitectura	Agosto - Septiembre	Configuración del *stack* Serverless/PostgreSQL, diseño UX/UI.
Fase II: Desarrollo del MVP Core	Septiembre - Octubre	Implementación de Autenticación, Lógica de Reservas (RF07), y Notificaciones (RF19).
Fase III: Seguridad, Pruebas y Entrega	Octubre - Noviembre	Implementación de RNF críticos (Seguridad, Consistencia RNF15) y Registro de Diagnóstico (RF16).

V.B. Análisis de Costos para la Microempresa (Enfoque Universitario)

Para un proyecto de microempresa, el **costo operativo real** se mantiene extremadamente bajo, usando **niveles gratuitos (Free Tiers)** de servicios en la nube. Sin embargo, para la sustentación de Ingeniería de Software, presento la **Proyección de Costos por Desarrollo**

Modular (Valoración Teórica) que refleja el costo si este MVP fuera desarrollado profesionalmente por un equipo externo, desglosando el valor por cada funcionalidad clave. Este análisis nos da una perspectiva real del valor de los módulos implementados, con un costo final en Pesos Colombianos (COP) más acorde a un proyecto de esta escala.

Módulo / Funcionalidad	Costo estimado (mínimo COP)	Costo estimado (máximo COP)
Diseño web responsivo	800000	1200000
Sistema de inicio de sesión / registro	700000	1000000
Gestión de productos	1000000	1500000
Chatbot personalizado	1200000	2500000
Agenda de citas	800000	1200000
Redirección a WhatsApp	400000	800000
Mensajes automáticos Gmail	500000	900000
Panel administrativo	1000000	1800000

Módulo / Funcionalidad	Costo estimado (mínimo COP)	Costo estimado (máximo COP)
Hosting y dominio (anual)	200000	400000
TOTAL	**6600000**	**11300000**

VI. Hoja de Ruta y Avances Críticos (Fase II)

Los avances se centran en el desarrollo del *core* de agendamiento. La Fase II, posterior a la entrega de noviembre, debe enfocarse en la sostenibilidad y el cumplimiento legal.

VI.A. Mejoras Críticas y Mitigación de Riesgos Arquitectónicos

Mi enfoque para la Fase II es demostrar la solidez técnica implementando las siguientes soluciones de ingeniería:

1. **Garantía de Consistencia (Optimistic Locking):** Implementar el patrón de **Optimistic Locking** en la tabla de citas de PostgreSQL. Esto es vital para manejar el agendamiento concurrente y asegurar la integridad de la agenda (RNF15), verificando si un registro fue modificado por otra transacción antes de confirmar la cita.

2. **Solución para Escalabilidad (Connection Pooling):** Para asegurar que el sistema escale correctamente (RNF05) sin saturar PostgreSQL, se recomienda la implementación de una capa de *Connection Pooling* (como Amazon RDS Proxy). Esta técnica permite a las funciones Serverless reutilizar conexiones existentes en lugar de crear nuevas constantemente, lo que reduce la latencia.

VI.B. Requerimientos de Cumplimiento Regulatorio y Ético

El manejo de datos de diagnóstico (RF16) me obliga a cumplir con la Ley 1581 de 2012 de Colombia :

- **Consentimiento Informado (RF01):** Debo integrar un mecanismo para capturar el **Consentimiento Informado explícito** del paciente para el tratamiento de sus datos de salud, según lo exige el Artículo 9 de la Ley 1581.

- **Ética en la IA:** La implementación del Asistente Virtual (Gemini) debe ser **culturalmente sensible**. Es crucial que el modelo sea *afinado* (*fine-tuned*) con el conocimiento específico del Taita, para evitar sesgos y garantizar que el soporte brindado sea preciso y ético.

VI.C. Continuación del Proyecto (Expansión)

1. **Monetización Segura:** Integrar pasarelas de pago colombianas (como **ePayco** o PayU) para la venta de productos, asegurando que toda la lógica sensible se ejecute **exclusivamente en el Backend Serverless** por seguridad.

2. **Gestión de Contenido Escalable:** Para el catálogo de productos y el contenido ancestral, planeo migrar la gestión de contenido estático a un **Headless CMS** (como Strapi), que es la mejor práctica en la arquitectura JAMstack para la escalabilidad de contenido.

VII. Conclusión

El Consultorio del Taita Jajoy es un proyecto que demuestra cómo la Ingeniería de Sistemas puede ofrecer soluciones de vanguardia (JAMstack + Serverless) con un alto impacto social, manteniendo un costo operativo viable para una microempresa.

El éxito del proyecto para la entrega de noviembre se basa en la implementación eficiente del MVP. Mi trabajo futuro se enfocará en asegurar la resiliencia técnica (escalabilidad, consistencia) y el cumplimiento ético-legal (Ley 1581), demostrando una comprensión integral del ciclo de vida del software, desde el diseño arquitectónico hasta la operación en un contexto real.

Referencia bibliográfica

Refactoring.guru. (2024). *Clasificación de los patrones de diseño*. https://refactoring.guru/es/design-patterns/classification