CHIBCHAWEB HOSTING PLATFORM
APLICACIÓN PARA UNA COMPAÑÍA DE HOSPEDAJE DE PÁGINAS WEB

DOCUMENTO DE DISEÑO Y ARQUITECTURA

Grupo A

Ingenieros:
Andrés José Acevedo Ardila
Brayan Steven Sánchez Casilima
Christian Camilo Lancheros Sánchez
Juan Esteban Ordoñez Sandoval
Karen Tatiana Bravo Rodríguez
Luis Felipe Mayorga Tibaquicha

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería
Ingeniería de sistemas

Bogotá D.C., Agosto 2025

ChibchaWeb
Hosting Platform

Grupo A

Documento de Diseño y Arquitectura

2025

Índice

1 Listado de Figuras

Índice de figuras

2 Listado de Tablas

Índice de cuadros

3 Descripción del Documento

3.1 Propósito y Audiencia

Este documento tiene como propósito describir la arquitectura de software propuesta para el desarrollo de una solución integral que apoye los procesos operativos de ChibchaWeb, una compañía dedicada al hospedaje de páginas web con sede en Sugamuxi y proyección de expansión internacional.
El documento presenta los componentes clave del sistema, los requerimientos funcionales y no funcionales, los patrones de diseño seleccionados, y los modelos arquitectónicos que sustentan la solución. También detalla la metodología de desarrollo adoptada, las decisiones técnicas tomadas y su justificación.
Este documento está dirigido a los siguientes perfiles:

- Equipo de Desarrollo: Para comprender la arquitectura, los patrones y las tecnologías definidas que guiarán la implementación del sistema.
- Clientes y Stakeholders de ChibchaWeb: Para validar que los requerimientos funcionales y no funcionales del negocio han sido entendidos y serán satisfechos por el sistema.
- Equipo de QA y Soporte Técnico: Para conocer los aspectos técnicos relevantes de la solución y planificar las estrategias de prueba y soporte.
- Líderes de Proyecto y Gerencia: Para tomar decisiones informadas sobre tiempos, recursos y alineación con objetivos estratégicos.

3.2 Organización del Documento

Este documento empieza con un panorama general del proyecto consistente en objetivos generales y específicos, seguidos de una contextualización del entorno y conceptos importante del proyecto, esto con el fin de generar un marco teórico que aclare y mejore la organización empresarial.
Luego de esto, se define el modelo de desarrollo y por lo tanto la organización del equipo de desarrollo. Después se tiene en cuenta las motivaciones y los riesgos relacionados con el desarrollo del proyecto, con el fin de estimar la utilidad. Además de las restricciones (tecnológico, negocio) y la gestión de calidad (con parámetros medibles para cumplir objetivos).
Por último, se definen los diagramas principales para empezar el proceso del producto, describiendo los diferentes modelos y entidades del desarrollo. Para terminar, se hace una pequeña retroalimentación con el fin de reconocer errores dentro del desarrollo del proyecto y corregirlos en próximos proyectos.

3.3 Convenciones

Identificadores de requerimientos:

- Requerimientos funcionales: RF-XX (ej. RF-01).

3.4 Terminología y Definiciones

- Cliente: Usuario registrado que adquiere servicios de hosting o dominios.
- Distribuidor: Usuario que intermedia la venta de dominios y recibe comisiones.
- Empleado: Usuario interno con funciones operativas dentro de ChibchaWeb.
- Administrador: Usuario con permisos totales de gestión.
- Ticket: Registro de solicitud de soporte técnico.
- Dominio: Nombre único utilizado para identificar un sitio web en Internet.
- Carrito: Módulo de compra donde se almacenan temporalmente productos antes del pago.
- Scrum: Marco de trabajo ágil usado para el desarrollo del proyecto.
- Frontend: Parte de la aplicación con la que interactúa el usuario final.
- Backend: Parte de la aplicación que maneja la lógica de negocio y la comunicación con la base de datos.
- PSE: Plataforma de pagos electrónicos utilizada en Colombia.
- XML: Lenguaje de marcado utilizado para intercambio de información estructurada.
- Railway: Plataforma de despliegue utilizada para alojar el sistema.
- Scraping: Técnica para extraer información de páginas web de terceros.
- Método de pago: Forma registrada por el usuario para realizar transacciones (tarjeta de crédito, PSE, etc.).

4 Generalidades del proyecto

4.1 Problema por resolver

ChibchaWeb, una empresa de hospedaje de sitios web ubicada en Sugamuxi, ha experimentado un crecimiento constante en su base de clientes, principalmente en Colombia y países cercanos, y se proyecta una expansión hacia otros continentes como África. Actualmente administra miles de sitios web bajo distintos paquetes de hosting (Chibcha-Platino, Chibcha-Plata y Chibcha-Oro) sobre plataformas Windows y Unix. Los clientes pueden escoger planes de pago flexibles (mensual, trimestral, semestral o anual), y aunque el método principal de pago es con tarjeta de crédito, se planea habilitar próximamente otros medios como PSE.

El modelo de negocio incluye además una red de distribuidores categorizados como BÁSICOS o PREMIUM, con diferentes esquemas de comisiones. ChibchaWeb también intermedia en el registro y transferencia de dominios a través de terceros registradores, ofreciendo este servicio adicional a sus clientes. A medida que el volumen de operaciones crece y los procesos se diversifican, la compañía enfrenta varios desafíos operativos y administrativos.

Actualmente, ChibchaWeb carece de una solución unificada que automatice y centralice sus procesos comerciales, operativos y de atención al cliente. Esto genera ineficiencias en la gestión de perfiles, pagos, comisiones, atención técnica y comunicación con registradores externos. La falta de un sistema integral limita la capacidad de escalar operaciones, introducir nuevos métodos de pago, y garantizar trazabilidad y control sobre los procesos internos.

4.2 Descripción general del sistema a desarrollar

El sistema a desarrollar para ChibchaWeb es una plataforma integral de gestión de servicios de hospedaje web, diseñada para automatizar, centralizar y optimizar los procesos operativos clave de la compañía. Esta solución tecnológica permitirá gestionar clientes, empleados, distribuidores, dominios, pagos y soporte técnico de manera eficiente, segura y escalable.

El sistema contará con módulos para la creación y administración de perfiles de clientes y empleados, y almacenará la información en una base de datos relacional optimizada incluyendo la validación de información personal y financiera garantizando la integridad evitando ciclos en esta; buscar y adquirir dominios mediante un carrito; adquirir paquetes de hosting.

El sistema contará con roles diferenciados por tipo de cuenta; búsquedas y consultas detalladas de todos los actores del sistema, así como herramientas para la generación de archivos de transferencia electrónica y seguimiento de actividades internas del proyecto; gestión de solicitudes de dominios con generación de archivos XML para su envío a registradores externos; Cálculo automático de comisiones a distribuidores según su categoría

(BÁSICO o PREMIUM); y un módulo de soporte técnico basado en tickets que garantiza el seguimiento y resolución estructurada de problemas reportados.

La solución incluirá una interfaz web en React, backend en FastAPI y base de datos en MySQL. Se aplicarán buenas prácticas de diseño para mantener una l ógica de negocio clara y una estructura coherente en el modelo de datos.

4.3 Objetivos de la solución

4.3.1 Objetivo general

- Desarrollar un sistema web que permita registrar, gestionar y monitorear cuentas de usuarios, adquisición de dominios y facturación de manera eficiente y centralizada.

4.3.2 Objetivos específicos

- Diseñar una base de datos relacional normalizada y sin ciclos innecesarios para soportar la lógica del sistema.
- Desarrollar una interfaz web funcional y responsiva utilizando React y Bootstrap para facilitar la interacción del usuario con el sistema.
- Construir un backend en FastAPI que gestione la lógica del negocio, la autenticación de usuarios y la comunicación con la base de datos.

4.4 Stakeholders

Stakeholder	Descripción
Cliente	Persona que encargó el proyecto y financia económicamente el desarrollo de la aplicación web.
Desarrolladores	Grupo encargado de desarrollar el código y las interfaces necesarias para el proyecto.
Usuarios finales	Posibles compradores interesados en adquirir dominios web.
Mantenedores	Desarrolladores encargados de actualizar el código y la página.
Product owner	Encargado de gestionar los requerimientos y verificar su cumplimiento/correcto desarrollo.

Cuadro 1: Stakeholders

5 Descripción y justificación de la metodología

Para el desarrollo del sistema de información de ChibchaWeb, se ha seleccionado la metodología ágil Scrum. Esta elección se basa en las características del proyecto y las necesidades del cliente, que requieren flexibilidad, entregas parciales funcionales y un involucramiento constante de los interesados en el proceso.

5.1 Scrum

Scrum es un marco ágil de desarrollo iterativo e incremental que permite entregar valor de manera temprana y continua a través de sprints (iteraciones de tiempo fijo, típicamente de 1 a 4 semanas).
En Scrum, el trabajo se organiza en:

- **Product Backlog:** Lista priorizada de funcionalidades y requerimientos del sistema.
- **Sprint Backlog:** Conjunto de tareas seleccionadas para el Sprint en curso.
- **Sprint Review y Sprint Retrospective:** Evaluaciones al finalizar cada sprint, enfocadas en la mejora continua.

5.1.1 Roles en Scrum aplicados al proyecto

- **Product Owner:** Representa los intereses de ChibchaWeb. Define y prioriza el Product Backlog.
- **Scrum Master:** Facilita el marco de trabajo Scrum, elimina impedimentos y promueve buenas prácticas.
- **Development Team:** Grupo autoorganizado de desarrolladores encargados de diseñar, codificar, probar e implementar el sistema.

5.2 Justificación de su uso en este proyecto

- **Adaptabilidad a cambios de requisitos:** Dado que el negocio de hospedaje web evoluciona rápidamente (nuevas formas de pago, expansión internacional), Scrum permite adaptarse con agilidad.
- **Entrega continua de valor:** Con cada sprint, se entregan versiones funcionales del sistema que pueden ser probadas por el cliente.
- **Comunicación constante con el cliente:** La retroalimentación frecuente mejora la alineación entre lo que se construye y lo que el cliente realmente necesita.
- **Reducción del riesgo:** La entrega incremental permite detectar errores o desviaciones de manera temprana.

- **Fomento de la colaboración y mejora continua:** Las reuniones retrospectivas y la autoorganización del equipo fortalecen la eficiencia del proceso.

6 Especificación y Recabación de Requerimientos Funcionales

La presente sección describe los requerimientos funcionales necesarios para el desarrollo de la aplicación de software para ChibchaWeb, con el fin de automatizar y optimizar los procesos de gestión de clientes, pagos, dominios, distribuidores y soporte técnico.

Código	Requerimiento Funcional
RF-01	El sistema debe permitir el registro de clientes con verificación por correo electrónico.
RF-02	El sistema debe permitir buscar dominios disponibles sin iniciar sesión.
RF-03	El sistema debe permitir adquirir un paquete de hosting a los clientes.
RF-04	El sistema debe permitir al cliente administrar el paquete de hosting que adquiera.
RF-05	El sistema debe permitir a clientes y distribuidores agregar dominios al carrito de compras.
RF-06	El sistema debe permitir a clientes y distribuidores adquirir dominios.
RF-07	El sistema debe permitir a clientes y distribuidores ver los dominios adquiridos.
RF-08	El sistema debe calcular el costo total del carrito automáticamente.
RF-09	El sistema debe permitir eliminar productos del carrito.
RF-10	El sistema debe enviar facturas al correo electrónico del usuario cuando realizan una compra.
RF-11	El sistema debe permitir transferir dominios a otros usuarios registrados en el sistema.
RF-12	El sistema debe permitir la creación de tickets de soporte para clientes y distribuidores.
RF-13	El sistema debe mostrar una notificación que indique transacciones exitosas o fallidas.
RF-14	El sistema debe permitir a cualquier usuario editar su propio perfil.
RF-15	El sistema debe permitir a los clientes y distribuidores agregar un método de pago.
RF-16	El sistema debe permitir a los clientes y distribuidores ver sus métodos de pago.
RF-17	El sistema debe permitir el registro de distribuidores con verificación por correo electrónico.
RF-18	El sistema debe permitir comprar dominios a los clientes y distribuidores.
RF-19	El sistema debe permitir transferir dominios a otros usuarios registrados en el sistema.
RF-20	El sistema debe permitir la creación de tickets de soporte para clientes y distribuidores.
RF-21	El sistema debe permitir al distribuidor consultar cuánta comisión ha recibido en todas sus compras.

Cuadro 2: Requerimientos Funcionales, parte 1.

Código	Requerimiento Funcional
RF-22	El sistema debe permitir al distribuidor ver el descuento de comisión que recibiría antes de comprar los dominios.
RF-23	El sistema debe permitir el registro de empleados con verificación por correo electrónico.
RF-24	El sistema debe permitir al empleado responder tickets asignados.
RF-25	El sistema debe permitir al administrador aprobar cuentas de empleados postulados.
RF-26	El sistema debe permitir al administrador asignar un nivel de soporte técnico a los empleados.
RF-27	El sistema debe permitir ver al administrador estadísticas de las ventas.
RF-28	El sistema debe permitir al administrador asignar un nivel de soporte técnico a los coordinadores.
RF-29	El sistema debe permitir al administrador ver todos los tickets.
RF-30	El sistema debe permitir al administrador filtrar los tickets por estados y niveles.
RF-31	El sistema debe permitir al administrador modificar precios de dominios por extensión.
RF-32	El sistema debe permitir al administrador modificar comisiones de los distribuidores.
RF-33	El sistema debe permitir al administrador gestionar usuarios (clientes, distribuidores y empleados): buscar, editar y eliminar cuentas.
RF-34	El sistema debe permitir al administrador gestionar paquetes de hosting: crear, editar y eliminar paquetes.
RF-35	El sistema debe permitir al coordinador ver los tickets sin asignar, en proceso y finalizados.
RF-36	El sistema debe permitir al coordinador asignar o escalar tickets a los empleados.
RF-37	El sistema debe permitir iniciar sesión según el tipo de usuario.
RF-38	El sistema debe mostrar los datos del perfil al iniciar sesión.
RF-39	El sistema debe permitir a cualquier usuario editar datos básicos de su perfil.
RF-40	El sistema debe mostrar una notificación que indique transacciones exitosas o fallidas.
RF-41	El sistema debe permitir ingresar una idea de negocio y mostrar opciones de dominios disponibles utilizando inteligencia artificial.
RF-42	El sistema debe permitir al usuario interactuar con un bot para resolver problemas generales.
RF-43	El sistema debe mostrar alternativas de dominios al usuario si el que buscó no está disponible.
RF-44	El sistema debe impedir realizar compras sin tener un método de pago registrado.
RF-45	El sistema debe permitir al usuario modificar y recuperar su contraseña

Cuadro 3: Requerimientos Funcionales, parte 2.

7 Motivadores Arquitecturales

7.1 Motivadores de Negocio

Los motivadores de negocio que se mostrarán a continuación definen las razones por las cuales ChibchaWeb requiere una nueva solución de software. Entre las cuales se encuentran:

7.1.1 Expansión internacional del negocio

ChibchaWeb, actualmente con clientes en Colombia y países vecinos, proyecta una expansión hacia mercados internacionales como África. Esto requiere una plataforma escalable y flexible que soporte la gestión de clientes, dominios y facturación en diferentes regiones.

7.1.2 Automatización de procesos clave

El sistema busca automatizar la creación, edición y eliminación de perfiles de clientes, empleados y distribuidores, reduciendo la carga operativa y los errores humanos, y facilitando la trazabilidad de operaciones críticas.

7.1.3 Mejora en la atención al cliente y soporte técnic

Al permitir el registro y seguimiento de tickets, el sistema mejorará el proceso de resolución de problemas, permitiendo al equipo de soporte priorizar incidencias y mejorar los tiempos de respuesta, alineándose con los niveles de servicio ofrecidos.

7.1.4 Gestión y segmentación de distribuidores

Con la existencia de distribuidores BÁSICOS y PREMIUM, el sistema debe permitir su categorización, cálculo de comisiones correspondientes (10

7.1.5 Flexibilidad en métodos de pago

Actualmente se aceptan tarjetas de crédito, pero se planea incluir otros métodos como PSE. Esto requiere una arquitectura extensible que permita integrar fácilmente nuevos métodos de pago en el futuro sin afectar la operatividad actual.

7.2 Restricciones de Tecnología

El desarrollo de la plataforma para ChibchaWeb estará sujeto a las siguientes restricciones tecnológicas, derivadas de decisiones de arquitectura, compatibilidad, mantenibilidad y recursos disponibles:

Tecnologías definidas por el equipo de desarrollo

7.2.1 Frontend

Se utilizará React.js como framework principal para la construcción de la interfaz de usuario, debido a su eficiencia, modularidad y gran comunidad de soporte.

7.2.2 Backend

Se empleará FastAPI con Python, por su rendimiento, facilidad de definición de servicios RESTful, y compatibilidad con documentación automática.

7.2.3 Base de datos

Se usará MySQL, una base de datos relacional robusta y ampliamente soportada, ideal para mantener la integridad referencial y relaciones entre entidades complejas (clientes, empleados, dominios, etc.).

7.2.4 Control de versiones y colaboración

El proyecto estará versionado mediante GitHub, lo cual permitirá control de cambios, trabajo colaborativo y despliegue continuo si se requiere.

7.3 Restricciones de Negocio

El desarrollo y funcionamiento del sistema estarán sujetos a diversas restricciones impuestas por las necesidades, normativas y condiciones del entorno comercial del proyecto. Estas restricciones afectan la manera en que se diseña y entrega la solución, e incluyen:

7.3.1 Políticas de manejo de cuentas

Solo se permitirán 4 tipos de cuentas: cliente, administrador, empleado y distribuidor, cada una con permisos y accesos claramente definidos. No se admitirán roles personalizados fuera de estas categorías durante la fase inicial del sistema.

7.3.2 Manejo de dominios basado en disponibilidad externa

La compra de dominios dependerá de su disponibilidad en registros externos. Si el dominio ya está tomado, no podrá ser adquirido ni facturado, lo cual impone una dependencia con proveedores externos y puede requerir lógica de validación en tiempo real.

7.4 Atributos de Calidad

Durante el desarrollo del sistema ChibchaWeb, se identificaron ciertos atributos de calidad que orientaron decisiones técnicas y de arquitectura. A continuación, se describen los principales, en función de la experiencia práctica con el sistema ya desplegado y en funcionamiento.

7.4.1 Disponibilidad

El sistema actualmente está desplegado mediante Railway, por lo tanto, su disponibilidad depende del servicio que ofrece esta plataforma. No se cuenta con una infraestructura propia, pero la integración entre base de datos, backend y frontend se mantiene activa y estable mientras Railway esté operativo. Esta disponibilidad es suficiente para los fines actuales, aunque si en un futuro fuera necesario se podría considerar una solución con mayor control sobre el entorno de producción.

7.4.2 Seguridad

Se implementó un sistema de autenticación por correo electrónico, en el cual los usuarios reciben un código de verificación. Solo quienes completen este proceso pueden acceder a las funciones reservadas para clientes. Esta medida ha resultado efectiva para restringir accesos y proteger la información del usuario, y se aplican validaciones en los formularios.

7.4.3 Rendimiento

La aplicación responde de forma fluida gracias a la conexión directa con las APIs del backend. No se ven demoras significativas al realizar operaciones como búsquedas, registros, etc.

7.4.4 Escalabilidad

Actualmente el sistema permite operaciones como búsqueda de dominios y la “compra” como tal, lo cual es una base para un entorno escalable. Aunque todavía no se ofrecen servicios de hosting reales desde la plataforma, el sistema está preparado para incluir esa funcionalidad más adelante, como tal el nucleo del sistema nos permite esto

7.4.5 Mantenibilidad

El código está organizado de forma clara, permitiendo modificar o añadir funcionalidades sin afectar lo ya implementado. Se han realizado ajustes menores y correcciones sin mayores inconvenientes, lo que demuestra que el sistema es mantenible. Además, se pueden actualizar componentes del frontend o backend de manera independiente.

7.4.6 Usabilidad

Desde la perspectiva del cliente, la interfaz es bastante intuitiva. Las funciones están bien distribuidas y no es necesario tener conocimientos técnicos para navegar o realizar operaciones básicas. En el caso del personal de soporte o empleados, hay más funciones que requieren cierta formación, como el manejo de tickets, pero están diseñadas de forma clara.

7.4.7 Interoperabilidad

El sistema se comunica con servicios externos mediante peticiones web, especialmente para consultar la disponibilidad de dominios. Para ello, se implementó un proceso de scraping que permite validar si un dominio está disponible o no.

7.4.8 Recabación y justificació de Requerimientos No Funcionales

Durante la planificación y construcción del sistema ChibchaWeb, se identificaron diversos requerimientos no funcionales a partir de las necesidades específicas del proyecto, la experiencia con despliegues reales, y las condiciones técnicas impuestas por el entorno en el que opera la aplicación. Estos requerimientos no son visibles directamente para el usuario final, pero son determinantes para que el sistema sea confiable, seguro y sostenible en el tiempo.

- Las operaciones realizadas por el usuario (como la búsqueda de dominios, la carga de formularios o la visualización de datos) no es de un tiempo significativo o excesivo, esto mejora la experiencia del usuario, esto gracias al uso de las Apis
- Aunque actualmente se encuentra todo en el servicio de railway para el despligue, fuera de eso el sistema es funcional y estable.
- Solo los usuarios verificados deben tener acceso a funcionalidades sensibles del sistema. Además, los datos sensibles como contraseña están protegidos
- El sistema esta preparado para soportar un aumento progresivo de usuarios y funcionalidades sin comprometer el rendimiento, aunque recalcar que al menos en esta fase la disponibilidad no depende tanto de nosotros, sino del servicio de railway
- En diferentes momentos del proyecto se han realizado cambios o mejoras con relativa facilidad, lo que indica que se ha logrado un diseño mantenible.
- Ya se implementó una primera forma de integración con servicios de disponibilidad de dominios usando scraping. Esta capacidad de interactuar con sistemas externos es la clave para estas funcionalidades.

7.4.9 Escenarios de Calidad

Los escenarios de calidad permiten evaluar cómo se comporta el sistema ante situaciones específicas que afectan atributos no funcionales como el rendimiento, la seguridad o la disponibilidad. A continuación se presentan algunos escenarios relevantes que reflejan condiciones reales o esperadas para la plataforma ChibchaWeb:

Consulta masiva de dominios disponibles

- **Contexto:** Un distribuidor accede a la plataforma y realiza varias búsquedas consecutivas para verificar la disponibilidad de dominios para sus clientes.
- **Estimulación:** Se ejecutan 10 a 15 búsquedas en un corto periodo de tiempo.
- **Respuesta esperada:** El sistema podra responder a estas consultas en un tiempo no tan significativo, sin errores ni bloqueos.

Intento de acceso sin verificación de correo

- **Contexto:** Un usuario se registra pero no verifica su correo electrónico.
- **Estimulación:** Intenta acceder a las funcionalidades de cliente como compra de dominio o visualización del perfil.
- **Respuesta esperada:** El sistema bloquea el acceso y solicita la verificación de correo antes de continuar.

Actualización del backend sin detener el sistema completo

- **Contexto:** Se implementa una mejora en una API del backend.
- **Estimulación:** Se despliega el nuevo backend mientras la base de datos y el frontend siguen activos.
- **Respuesta esperada:** El sistema sigue funcionando normalmente o se interrumpe solo por un corto tiempo. No se generan inconsistencias.

Interacción de un cliente no técnico

- **Contexto:** Un cliente nuevo accede al sistema para registrar su primer dominio.
- **Estimulación:** Usa la interfaz para buscar un dominio, registrar sus datos y simular un pago.
- **Respuesta esperada:** El cliente puede completar la operación sin necesidad de asistencia. Los pasos son claros y comprensibles.

Caída temporal del servicio Railway

- **Contexto:** Railway presenta una interrupción temporal.
- **Estimulación:** Usuarios intentan ingresar al sistema durante el tiempo en que el servicio está inactivo.

- **Respuesta esperada:** El sistema no responde (fuera de servicio), pero se recupera automáticamente cuando Railway vuelve a estar disponible, sin pérdida de información.

8 Contexto

8.1 Escenarios operacionales

Estos Escenarios operacionales representan las interacciones comunes o típicas que pasan entre los diferentes actores de la plataforma de ChibchaWeb, Cada escenario describe como un usuario o actor logra el objetivo deseado utilizando el sistema en un contexto especifico

8.1.1 Escenario 1: Registro y verificación de cuenta

- **Actor:** Usuario
- **Descripción:** Un usuario se registra en la plataforma ingresando sus datos personales. Luego, valida su cuenta a través de un token enviado por correo electrónico.
- **Resultado esperado:** El usuario puede iniciar sesión y acceder a las funciones de su rol.

8.1.2 Escenario 2: Compra de un paquete de hosting

- **Actor:** Cliente
- **Descripción:** El cliente agrega un método de pago, selecciona un paquete, realiza el pago y se activa el servicio.
- **Resultado esperado:** El sistema registra la compra.

8.1.3 Escenario 3: Registro de solicitud de dominio

- **Actor:** Usuario (Cliente o Distribuidor)
- **Descripción:** El usuario busca un dominio, valida su disponibilidad y envía una solicitud.
- **Resultado esperado:** La solicitud es procesada y almacenada para generación de archivos XML.

8.1.4 Escenario 4: Atención de ticket de soporte

- **Actor:** Usuario
- **Descripción:** El usuario crea un ticket describiendo su problema. Soporte lo atiende, lo puede escalar y responder. El administrador puede asignarlo.
- **Resultado esperado:** se le da solución al ticket lo más pronto posible

8.1.5 Escenario 5: Gestión administrativa de usuarios

- **Actor:** Administrador
- **Descripción:** Desde el panel de administración, el administrador puede gestionar precios, paquetes y consultar
- **Resultado esperado:** Los cambios se reflejan en la base de datos y actualizan el aplicativo en tiempo real

8.1.6 Escenario 6: Cálculo y pago de comisiones

- **Actor:** Administrador
- **Descripción:** El sistema calcula las comisiones para los distribuidores según su categoría básico o premium (recordar que esto es un descuento)
- **Resultado esperado:** El administrador tiene listos los valores y la documentación de soporte.

8.1.7 Escenario 7: Consulta de perfil personal

- **Actor:** Usuario
- **Descripción:** El usuario inicia sesión, accede a su perfil y visualiza información relevante como nombre, correo, y acceso a sus métodos de pago o registrar uno nuevo
- **Resultado esperado:** El usuario tiene acceso a un panel completo sobre sus datos de forma clara

8.1.8 Escenario 8: Transferencia de dominio vía correo

- **Actor:** Usuario
- **Descripción:** Un usuario puede transferir un dominio en su propiedad al correo de otro usuario.
- **Resultado esperado:** La transferencia queda registrada y se notifica a los usuarios involucrados

8.2 Casos de Uso: Descripción y Modelo

Los casos de uso del sistema ChibchaWeb describen las funcionalidades clave que pueden ser ejecutadas por los distintos actores, permitiendo alcanzar objetivos específicos mediante la interacción con la aplicativo. Este modelo facilita la comprensión de los límites del sistema, las responsabilidades de cada actor y las relaciones entre funcionalidades.

Actores Del Sistema:

- **Usuario:** Actor general del que heredan todos los demás.
- **Cliente:** Usuario registrado que puede adquirir servicios y registrar dominios.
- **Distribuidor:** Usuario que gestiona múltiples dominios y cobra a sus propios clientes.
- **Empleado:** Usuario interno con funcione operativas.
- **Soporte:** Subtipo de empleado, encargado de atender tickets.
- **Administrador:** Usuario con permisos totales para gestionar perfiles, dominios y comisiones

8.2.1 Resumen De Casos De Uso

Categoría	Caso de Uso	Actor(es)
Autenticación	Iniciar Sesión	Usuario
	Registrarse como Cliente	Cliente
	Registrarse como Distribuidor	Distribuidor
	Verificar Cuenta por Token	Usuario
Gestión de Perfil	Ver Perfil	Usuario
	Buscar Cliente, Distribuidor o Empleado	Administrador
	Gestionar Perfiles de Cliente	Administrador
	Gestionar Perfiles de Distribuidor	Administrador
	Gestionar Perfiles de Empleado	Administrador
	Registrar Empleado	Administrador
Métodos de Pago	Agregar Método de Pago	Cliente
	Realizar Pago	Cliente
Servicios	Comprar Paquete	Cliente
	Cobrar a sus Clientes	Distribuidor
Dominios	Registrar Solicitud de Dominio	Usuario
	Transferir Dominio por Correo	Usuario
	Generar Archivo de Dominio	Administrador
Soporte Técnico	Registrar Ticket de Soporte	Usuario
	Consultar Tickets	Soporte
	Atender y Escalar Tickets	Soporte
	Responder Ticket	Soporte
	Asignar Ticket	Administrador
Comisiones	Calcular Comisiones a Distribuidores	Administrador

Cuadro 4: Resumen De Casos De Uso

9 Descripción y justificación de patrones de diseño adoptados y arquitectura de software definida.

En el desarrollo de la plataforma ChibchaWeb se implementaron diversos patrones de diseño que permitieron mantener la claridad en la arquitectura, separación de responsabilidades y facilidad de mantenimiento del código.

9.0.1 Patrón DTO (Data Transfer Object)

Utilizado para la validación y transporte de datos entre el frontend y el backend. Se implementó mediante Pydantic en FastAPI, garantizando que la información recibida cumpla con el formato esperado antes de ser procesada, reduciendo así errores y mejorando la seguridad de entrada de datos.

9.0.2 Patrón Active Record

Aplicado en los modelos de datos mediante SQLAlchemy, permitiendo que cada entidad del sistema (Cliente, Dominio, Paquete, etc.) contenga tanto su representación como las operaciones CRUD asociadas, simplificando el acceso y manipulación de la base de datos.

9.0.3 Patrón Controller

Implementado a través de los routers de FastAPI, donde cada módulo del sistema cuenta con controladores que gestionan las solicitudes HTTP, orquestan la lógica de negocio y retornan las respuestas correspondientes.

9.0.4 Encapsulación de lógica en servicios

Funcionalidades específicas, como el generador de respuestas mediante inteligencia artificial, se desarrollaron como servicios independientes. En este componente se emplearon los patrones Strategy (para permitir distintas formas de generar respuestas según el contexto) y Adapter (para integrar proveedores externos de IA con la lógica interna del sistema).

9.0.5 Patrón Template Method

Los flujos de facturación y la generación de solicitudes XML para registradores externos se estructuraron siguiendo este patrón, definiendo la secuencia general de pasos y permitiendo que subclases concreten detalles específicos, lo que facilita cambios futuros sin modificar la estructura base del proceso.

En conjunto, la aplicación de estos patrones ha permitido desarrollar un sistema modular, escalable y adaptable a los cambios, manteniendo buenas prácticas de ingeniería de software y asegurando la coherencia entre los distintos componentes.

10 Puntos de Vista y Modelos Arquitecturales

10.1 Punto de Vista: Descripción del problema.

Para describir el problema a solucionar al desarrollar la plataforma propuesta, se presentan a continuación dos diagramas para describir el punto de vista de contexto de la solución:

10.1.1 Punto de Vista: Contexto de la solución

Donde se aprecia el sistema a desarrollar (ChibchaWeb Hosting Platfom) y las entradas y salidas que posee el sistema:

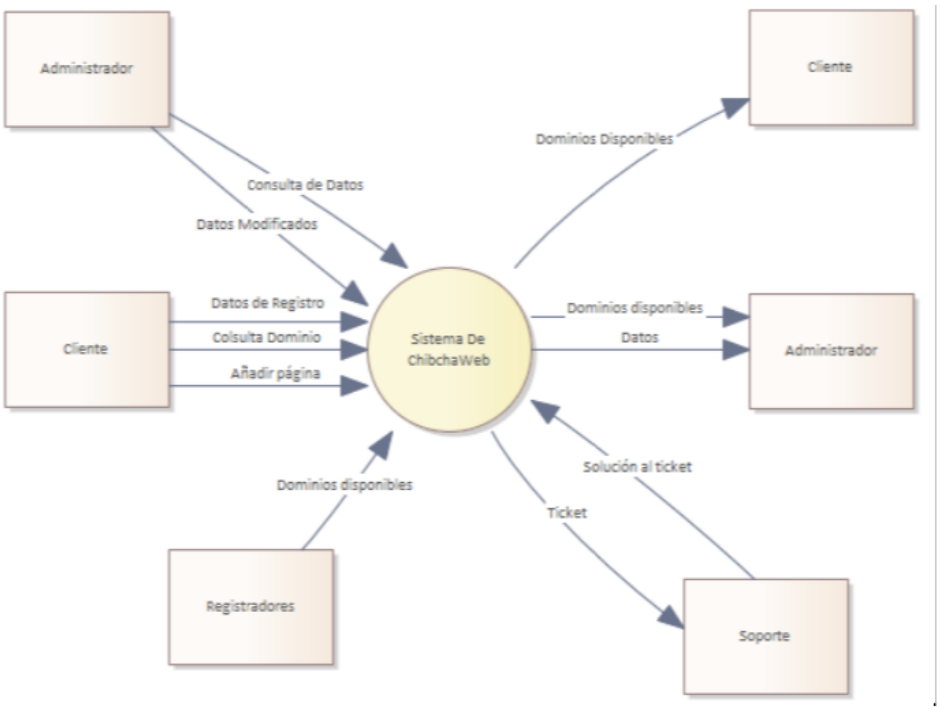


Figura 1: Diagrama de contexto.

10.2 Punto de Vista: Interacción

El punto de vista de interacción tiene como propósito representar cómo los diferentes actores del sistema ChibchaWeb se comunican con los componentes funcionales durante la ejecución de casos de uso específicos.

10.2.1 Diagrama de Casos de Uso

Muestra la interacción entre los actores principales del sistema ChibchaWeb y los casos de uso que pueden ejecutar. Esta vista representa la funcionalidad del sistema desde el punto de vista de cada actor y su relación directa con los servicios ofrecidos, recalcar que

aunque igual son numeros diagramas se intento ser lo más especifico en las funcionalidades planteadas para el proyecto, las funcionalidades de valor agregado al proyecto no fueron incluidas tales como buscar dominio por IA o el chatbot, el diagrama completo se encontrara en la sección de anexos.

10.2.2 Diagramas de Secuencia

Los diagramas de secuencia representan la dinámica del sistema durante la ejecución de casos de uso específicos. En ellos se detalla el orden cronológico de los mensajes intercambiados entre los actores y los objetos o componentes internos del sistema, todos estos diagramas se encontraran en la sección de anexos.

Esto Nos Permite Ver:

- Cómo se coordinan los distintos elementos para cumplir una funcionalidad.
- Qué actores o servicios están involucrados.
- Cómo fluye la lógica de negocio a través de las capas del sistema.

1. Agregar Método De Pago

Este diagrama de secuencia representa el proceso mediante el cual un cliente registra un nuevo método de pago (tarjeta de crédito o débito) en la plataforma ChibchaWeb. El proceso inicia cuando el cliente ingresa los datos de su tarjeta en el formulario correspondiente.

El sistema separa internamente el flujo en dos pasos:

1. Primero se realiza una llamada a la API /tarjeta para registrar los datos de la tarjeta en la base de datos.
2. Luego, se ejecuta la llamada a /MetodoDePagopara asociar el método de pago a la cuenta del cliente.

Ambas solicitudes son gestionadas por el módulo encargado de manejar los métodos de pago. Una vez completadas las operaciones en la base de datos, el sistema confirma al cliente que el método fue agregado correctamente.

2. Registrarse Como Distribuidor

Este diagrama de secuencia ilustra el proceso completo de registro de un nuevo distribuidor en la plataforma ChibchaWeb. El flujo inicia cuando el usuario llena el formulario de registro con su información personal y empresarial.

Una vez enviado el formulario:

1. El controlador de registros valida y procesa los datos recibidos.

2. Se genera una nueva cuenta en la base de datos del sistema.
3. Simultáneamente, el sistema genera un token de verificación y lo envía al correo electrónico del distribuidor mediante el servicio de correo.

El distribuidor recibe un mensaje en pantalla indicando que debe verificar su cuenta para completar el proceso de registro y poder acceder al sistema con normalidad.

3. Registrarse Como Cliente

Este diagrama de secuencia describe el proceso mediante el cual un cliente se registra en la plataforma ChibchaWeb. El flujo comienza cuando el cliente llena el formulario de registro desde el frontend.

El sistema ejecuta los siguientes pasos:

1. La información del formulario es enviada a la API de registro.
2. La API delega la validación de los datos a un servicio de validación.
3. Una vez validados los campos, la API crea una nueva cuenta en la base de datos.
4. Posteriormente, se genera un token de verificación a través del servicio de tokens, el cual es almacenado nuevamente en la base de datos.
5. Finalmente, se utiliza el servicio de correo para enviar el mensaje de verificación al correo del cliente.

El proceso concluye mostrando un mensaje al usuario con la instrucción de verificar su cuenta.

4. Generar archivo de dominio

Este diagrama de secuencia representa el proceso completo mediante el cual un usuario envía una solicitud para registrar un dominio a través de la plataforma ChibchaWeb. El flujo incluye desde el ingreso inicial de los datos hasta la generación y envío del archivo XML al registrador externo.

El proceso sigue los siguientes pasos:

1. El usuario llena el formulario y envía la solicitud.
2. El controlador de dominio envía los datos al servicio de dominio, que los procesa y gestiona.
3. La solicitud se guarda a través del backend API, que la inserta en la base de datos y recibe un identificador.
4. Luego, se genera un archivo XML con los datos de la solicitud.

5. El archivo generado es enviado al registrador externo.
6. Tras la confirmación del registrador, se actualiza el estado de la solicitud en el sistema.
7. Finalmente, se muestra al usuario un mensaje confirmando que la solicitud fue enviada correctamente.

5. Cargar Pagos a Tarjeta

Este diagrama de secuencia muestra el proceso que se ejecuta cuando un cliente confirma la compra de un servicio desde el carrito, utilizando una tarjeta previamente registrada como método de pago.

El flujo se desarrolla de la siguiente forma:

1. El cliente confirma la compra desde el frontend, lo que activa el controlador de pagos.
2. El controlador de pago solicita al backend API la verificación del método de pago seleccionado.
3. El backend consulta la base de datos para validar la tarjeta asociada y, si es válida, procede a procesar la transacción con el servicio de pago externo.
4. El servicio responde si el resultado fue aprobado o rechazado.
5. En caso de aprobación:
   (a) Se confirma el pago del carrito.
   (b) Se actualiza el estado del carrito en la base de datos.
   (c) El cliente recibe un mensaje confirmando el éxito de la operación.
6. Si el pago es rechazado:
   (a) El sistema muestra un mensaje de error indicando que la transacción fue fallida.

6. Registrar Solicitud De Dominio

Este diagrama de secuencia representa el flujo de interacción cuando un usuario desea registrar un dominio a través del sistema ChibchaWeb. La funcionalidad incluye una validación previa en tiempo real para comprobar si el nombre de dominio está disponible.

El proceso se detalla a continuación:

1. El usuario ingresa el nombre del dominio en el formulario de registro.
2. El controlador de registro de dominio consulta la disponibilidad del dominio utilizando un servicio de scraping, el cual accede a un registrador externo.

3. Si el dominio está disponible:
   (a) Se realiza una solicitud POST para registrar el dominio a través del backend API.
   (b) Esta solicitud se guarda en la base de datos, y se actualiza el estado del dominio como ocupado.
   (c) El sistema muestra un mensaje indicando que la solicitud fue registrada exitosamente.
4. Si el dominio no está disponible:
   (a) Se muestra un mensaje de error al usuario indicando que el dominio ya está ocupado.

7.Gestionar Perfiles De Clientes/Empleado

Este diagrama de secuencia representa el proceso mediante el cual un distribuidor puede buscar, editar o eliminar perfiles de clientes en la plataforma.

El flujo incluye:

- Búsqueda: El distribuidor consulta un cliente. El sistema recupera los datos desde la base de datos y los muestra en la interfaz.
- Edición: Se modifican los datos del perfil. La información se actualiza en la base de datos y se notifica al distribuidor.
- Eliminación: Se confirma la eliminación del perfil, la base de datos ejecuta la operación y se devuelve una respuesta de confirmación.

8. Calcular Comisión De Distribuidores

Este diagrama de secuencia representa el proceso mediante el cual un administrador ejecuta el cálculo de comisiones para distribuidores en función de sus ventas registradas.

El flujo general es el siguiente:

1. El administrador inicia el proceso desde la vista administrativa.
2. El controlador de comisiones solicita al servicio de comisiones la ejecución del cálculo.
3. Se consultan en la base de datos las ventas asociadas a cada distribuidor.
4. El sistema aplica el porcentaje de comisión correspondiente (10% para distribuidores BÁSICOS y 15% para PREMIUM).

9. Generar Transferencia

Este diagrama de secuencia describe el proceso mediante el cual un cliente transfiere la propiedad de un dominio a otro usuario de la plataforma, utilizando su dirección de correo como identificador.

El flujo se compone de los siguientes pasos:

1. El cliente ingresa el dominio a transferir y el correo de destino en la interfaz correspondiente.
2. El controlador de transferencia gestiona la solicitud y la envía al API de transferencia.
3. El backend consulta la base de datos para verificar que la cuenta de destino existe y que el dominio pertenece al cliente actual.
4. Si todo es válido, se actualiza el propietario del dominio en la base de datos.
5. Finalmente, se muestra al cliente un mensaje indicando que la transferencia fue exitosa.

10. Registrar Empleado

Este diagrama muestra el proceso mediante el cual un administrador registra un nuevo empleado en la plataforma ChibchaWeb.

El flujo sigue los siguientes pasos:

1. El administrador completa el formulario de registro desde la vista administrativa.
2. El controlador de registro de empleados recibe los datos y realiza una solicitud POST a la API de registro.
3. La API almacena la información en la base de datos y devuelve una confirmación.
4. Finalmente, el sistema muestra un mensaje notificando que el empleado fue registrado exitosamente.

11. Registrar Tickets

Este diagrama de secuencia representa el proceso mediante el cual un usuario crea un ticket de soporte en la plataforma ChibchaWeb, detallando un asunto y una descripción del problema.

El flujo es el siguiente:

1. El usuario completa el formulario de soporte con los datos requeridos.
2. El controlador de tickets procesa la solicitud y la envía mediante una petición POST a la API de creación de tickets.

3. La API almacena el ticket en la base de datos y genera un identificador único.
4. Finalmente, se muestra al usuario un mensaje de confirmación indicando que el ticket fue creado correctamente.

12. Consultar Tickets

Este diagrama de secuencia describe el proceso en el cual un usuario (cliente, distribuidor, empleado o soporte) accede al módulo de soporte técnico para consultar los tickets que le han sido asignados o que ha creado previamente.

El flujo contempla dos variantes según el tipo de usuario:

1. Si el usuario es cliente o distribuidor, se realiza una solicitud GET /consultarTicketporIDCUENTA.
2. Si es empleado o soporte técnico, se usa GET /mis-tickets-empleado.

En ambos casos:

- La solicitud es gestionada por el controlador, que la envía a la API de tickets.
- La API consulta la base de datos y devuelve la lista de tickets encontrados.
- Finalmente, la lista se muestra al usuario en la interfaz.

13. Atender Y Escalar Tickets

Este diagrama de secuencia ilustra el proceso que realiza un usuario del área de soporte técnico para visualizar sus tickets asignados, actualizar su estado (por ejemplo, de ”pendiente” a ”en proceso” o ”resuelto”) y, si corresponde, escalar el ticket a un nivel superior.

El flujo incluye dos acciones principales:

1. **Atender Ticket (Cambiar Estado):**
   - El soporte accede al panel donde visualiza sus tickets.
   - A través del controlador, el sistema obtiene los tickets desde la API y los recupera de la base de datos.
   - El soporte selecciona un ticket y un nuevo estado, lo cual genera una solicitud PATCH /CambiarEstadoTicket.
   - El estado se actualiza en la base de datos y se notifica al usuario.
2. **Escalar Ticket:**
   - Si el ticket requiere mayor atención, el soporte ejecuta una acción para escalarlo.
   - Se envía una solicitud PATCH /CambiarNivelTicket con el identificador del ticket.
   - El backend registra el cambio de nivel, actualizando la información del ticket y notificando al usuario que el ticket fue escalado.

14. Iniciar Sesión

Este diagrama de secuencia describe el proceso de inicio de sesión en la plataforma ChibchaWeb por parte de un usuario, contemplando tres posibles resultados: acceso completo, acceso limitado (por cuenta no verificada) o error por credenciales inválidas.

El flujo es el siguiente:

1. El usuario ingresa sus credenciales en la vista de login.
2. El controlador de login las envía a la API, que consulta la base de datos para verificar:
   - La existencia del usuario.
   - Que la contraseña coincida.
   - Si la cuenta está verificada.
3. Si las credenciales son válidas:
   (a) Si la cuenta está verificada: se inicia sesión, se devuelve un token y se redirige según el rol.
   (b) Si no está verificada: se limita el acceso y se notifica al usuario que debe verificar su cuenta.
4. Si las credenciales son inválidas:
   (a) Se muestra un mensaje de error indicando el fallo.

15.Buscar Cuenta

Este diagrama de secuencia muestra el proceso que sigue un administrador para consultar la información de una cuenta existente, utilizando como criterio de búsqueda un correo electrónico o ID.

El flujo es el siguiente:

1. El administrador ingresa el correo o identificador en la vista de administración.
2. El controlador de búsqueda de cuentas envía la solicitud al endpoint correspondiente de la API.
3. La API consulta la base de datos para recuperar los datos de la cuenta asociada.
4. Una vez encontrada, se retorna la información del perfil y se presenta al administrador.

16.

Este diagrama de secuencia describe el proceso mediante el cual un usuario accede a la sección "Mi Perfil" dentro de la plataforma ChibchaWeb para visualizar su información personal.

El flujo incluye los siguientes pasos:

1. El usuario navega a la vista de perfil desde la interfaz.
2. El controlador de perfil realiza una solicitud POST al endpoint /cuenta_por_correo en la API, pasando el identificador del usuario.
3. La API consulta la base de datos para obtener los datos asociados al correo.
4. Los datos del perfil (como nombre, correo, dirección, etc.) son devueltos y mostrados en pantalla.

17. Validar Datos

Este diagrama de secuencia describe el proceso de validación de datos ingresados por un usuario en formularios del sistema ChibchaWeb. Este flujo es común en procesos como registro, adición de tarjetas o dominios, y garantiza que la información ingresada cumpla con formato y disponibilidad antes de ser procesada.

El flujo contempla dos escenarios:

1. **Datos válidos:**
   (a) El usuario ingresa los datos en el formulario.
   (b) El frontend los envía al controlador de validación, que verifica el formato y la disponibilidad en la base de datos.
   (c) Si no hay errores, se devuelve una respuesta confirmando que los datos fueron validados correctamente.
2. **Datos inválidos o duplicados:**
   (a) En caso de errores de formato o duplicidad (por ejemplo, correo o dominio ya registrados), el sistema responde con una lista de errores que se muestran al usuario.

10.3 Punto de Vista: Desarrollo

En este diagrama, se aprecian tanto los componentes necesarios para el desarrollo del aplicativo, como su interacción con los demás, esto para poder tener una mayor comprensión del problema a solucionar.

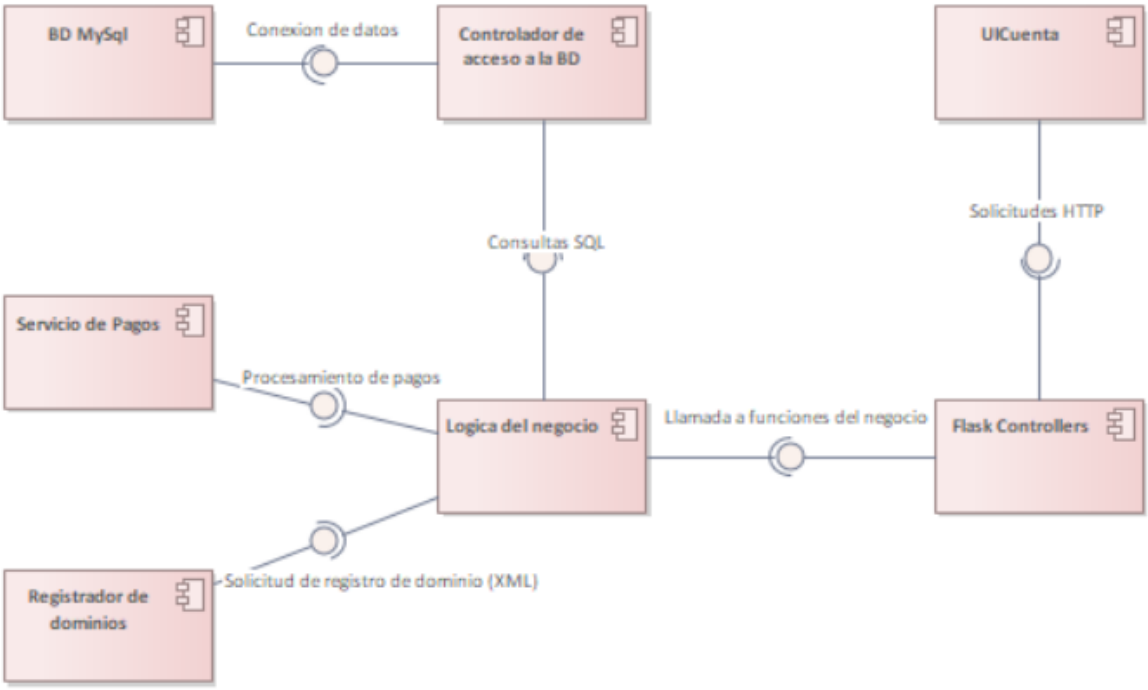


Figura 2: Diagrama de componentes

10.4 Punto de Vista: Modelo Relacional de la BBDD

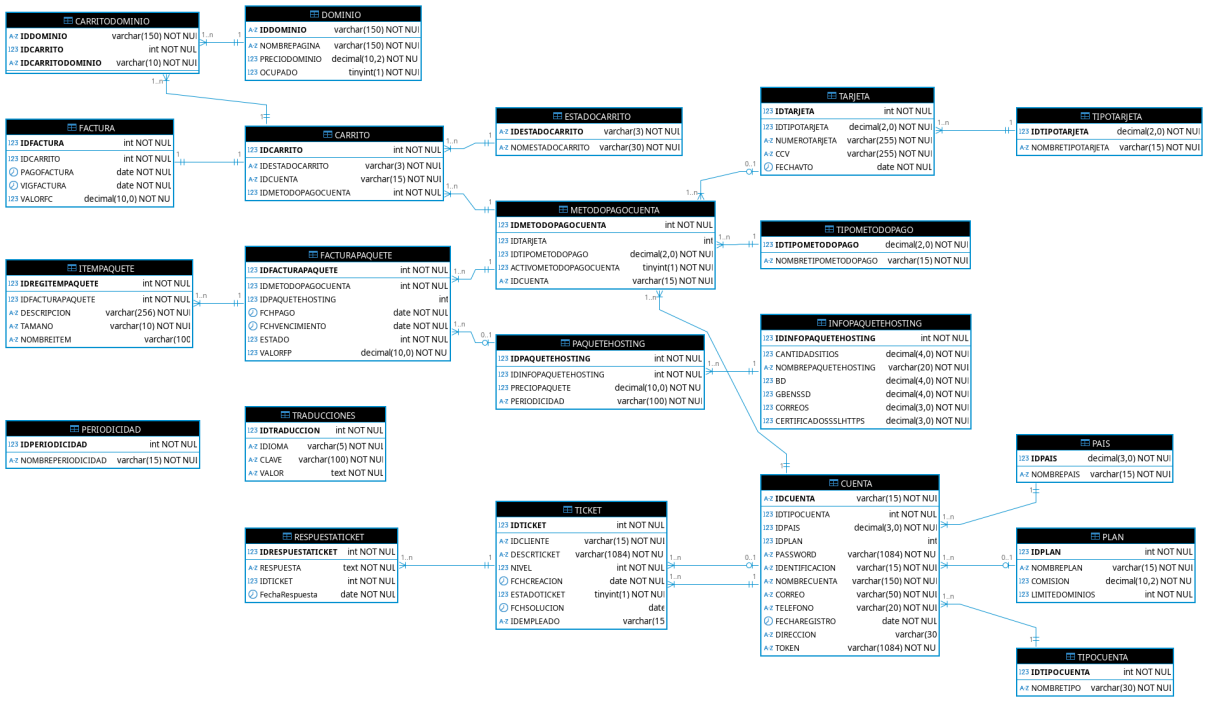


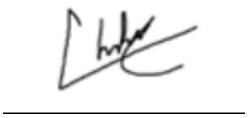
Figura 3: Diagrama de relacional.

11 Firmas de aceptación con fecha

Andrés José Acevedo Ardila

Brayan Steven Sánchez Casilima

Christian Camilo Lancheros Sánchez



Juan Esteban Oviedo Sandoval

Karen Tatiana Bravo Rodríguez

Luis Felipe Mayorga Tibaquicha

Firmado el día: 07 de Agosto del año 2025