



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO

**SISTEMA INTEGRAL DE GESTIÓN OPERATIVA WEB PARA
PEQUEÑAS Y MEDIANAS EMPRESAS – PANEL
ADMINISTRATIVO INTERNO.**

**TITULACIÓN INTEGRAL INFORME DE RESIDENCIA
PROFESIONAL**

**PARA OBTENER EL TÍTULO PROFESIONAL DE:
INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y
COMUNICACIONES**

PRESENTA:
DANITZA ANHEL RAMOS CHAVEZ

NOMBRE DEL ASESOR:
DR. JUAN RAMOS RAMOS



APIZACO, TLAX., FEBRERO 2026



Instituto Tecnológico de Apizaco
Subdirección de Planeación

Apizaco, Tlaxcala, 09/febrero/2026

Oficio No. S.C./OPV/0003/26

ASUNTO: **Liberación de proyecto
para titulación integral**

CECILIA YARELLI ARREGÍN LARA
JEFA DEL DEPTO. DE DIVISIÓN DE ESTUDIOS PROFESIONALES
P R E S E N T E.

POR ESTE MEDIO LE INFORMO QUE HA SIDO LIBERADO EL SIGUIENTE PROYECTO PARA LA TITULACIÓN INTEGRAL.

NOMBRE DEL EGRESADO:	RAMOS CHAVEZ DANITZA ANHEL
CARRERA:	INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
NO. DE CONTROL:	C20140952
NOMBRE DEL PROYECTO:	Sistema Integral de Gestión Operativa Web para Pequeñas y Medianas Empresas - Panel Administrativo Interno
PRODUCTO U OPCIÓN:	TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

AGRADEZCO DE ANTEMANO SU VALIOSO APOYO EN ESTA IMPORTANTE ACTIVIDAD PARA LA FORMACIÓN PROFESIONAL DE NUESTROS EGRESADOS.

ATENTAMENTE

*Excelencia en Educación Tecnológica.
Pensar para Servir, Servir para Triunfar®*

MIQUELINA SÁNCHEZ PULIDO
JEFA DEL DEPTO. DE SISTEMAS Y COMPUTACIÓN

ccp. Coordinación de carrera
Expediente
MSP/mrol*



SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL
DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

JUAN RAMOS RAMOS
ASESOR INTERNO



2026
año de
**Margarita
Maza**





Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Apizaco
División de Estudios Profesionales

Tzompantepec, Tlaxcala, 10/Febrero/2026

DANITZA ANHEL RAMOS CHAVEZ
EGRESADO(A) DE LA CARRERA DE INGENIERÍA
EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
CON NÚMERO DE CONTROL C20140952.
P R E S E N T E.

Por este medio, me permito informar a usted que por aprobación de la comisión revisora asignada para valorar el trabajo que mediante la opción **TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL** y cuyo tema es: "**SISTEMA INTEGRAL DE GESTIÓN OPERATIVA WEB PARA PEQUEÑAS Y MEDIANAS EMPRESAS - PANEL ADMINISTRATIVO INTERNO**", conforme a lo establecido en el procedimiento para la obtención del Título Profesional en el Instituto Tecnológico, la División de Estudios Profesionales a mi cargo le otorga la:

AUTORIZACIÓN DE IMPRESIÓN

Debiendo entregar **1 USB** del mismo, apropiadamente empastados, a la brevedad, para presentar su Acto de Recepción Profesional.

Sin otro particular por el momento me despido.

ATENTAMENTE

Excelencia en Educación Tecnológica®

Pensar para Servir, Servir para Triunfar®


CECILIA YARELY ARREGUÍN LARA
JEFA DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES



ccp. Archivo
CYAL/Gsc*



2026
año de
Margarita
Maza



Agradecimientos

La vida está llena de logros y fracasos. En cada uno de ellos vivimos, experimentamos y aprendemos. Todo camino es difícil, lleno de baches y piedras que a veces nos impide el paso, pero cuando te das cuenta en la posición en la que te encuentras, entiendes que cada sacrificio y lágrima valió la pena. Que a pesar de los obstáculos fuiste valiente, y te arriesgaste. Incluso esas veces que dudabas de ti y de tu alrededor, simplemente seguiste, con miedo, pero decidido a dar ese salto de fé.

Hoy la vida me da una gran prueba de lo valiente que he sido, y me ha recompensado con un valioso y glorioso logro. Parte de ello, es importante agradecerte a ti mismo y a los que siempre te apoyaron en los momentos difíciles como aquellos llenos de diversión. Primero que nada, quiero agradecer a mi cuerpo y mente, por siempre seguir aun en los días en los que no tenía ánimos. Incluso en aquellos en los que me sentía verdaderamente completa. Tú y nadie más que tú, sabes sobre tus cataratas. Agradezco lo fuerte que fuiste a pesar del miedo, los obstáculos, los errores y las dudas. Por siempre tener en mente lo importante que era para ti cumplir tus objetivos a pesar de todo. Y aquí estas. En lo que es uno de los muchos grandes logros que te prometiste cumplir.

De igual forma, quiero agradecer a mis padres Mireya Chavez Barrios y Armando Ramos Flores. Por enseñarme que camino quiero tomar. Por darme las herramientas necesarias para ser una persona objetiva. Entender lo que es hacer bien las cosas, lograr identificar cuando me equivoco y no simplemente quedarse estancado. Ser resiliente, ser fuerte, cumplir tus sueños a pesar de todo aquello que en ocasiones suele frenarte. Entender que son solo eso, obstáculos, pero no impedimentos para lograr lo que siempre soñaste. Por enseñarme lo que es cada uno de ustedes, tomar lo bueno y lo malo como un ancla a aquello que sí quiero ser en la vida. Aprender de ustedes ha sido uno de los más grandes tesoros que he tenido en la vida.

Quiero agradecer a mis hermanos, Verónica Mireya Ramos Chavez, Magali Detzani Ramos Chavez y Erick Genovanni Ramos Chavez. Por ser mis cómplices y juzgadores de primera, de forma constructiva, por supuesto. Por convertirse en mis joyas preferidas del universo. Compartirme sus experiencias, sus miedos, gustos y esencia. Cada uno de nosotros cuatro somos completamente distintos, pero complementa uno al otro de maneras inimaginables. Serán mis compañeros de vida por el resto de ella.

Así mismo, agradezco a mis amigos Belen Yerahí Ramírez Suarez, Lorena Aguilar Carvajal, Monserrat Marín Vázquez, Jenny Palacios Palacios, Joel Huerta Cabrera, Hiram Montiel Ramírez y Carlos Alberto Hernández Sastre, por enseñarme el significado de la amistad y el amor en toda su expresión. Apoyarme en cada paso, compartir palabras de aliento, ánimo y reflexión. Ser ese impulso que a veces necesitaba para seguir adelante. Me hicieron recordar lo valiente que he sido y la valiosa persona en la que me he convertido. Gracias por compartir lágrimas y risas. Ser mis compañeros de trabajo, destacando que cada uno de nosotros somos importantes dando nuestro granito de arena en todo lo que hacemos juntos. Ustedes hicieron que la universidad fuera una de las experiencias más maravillosas que he tenido.

A Diego Alberto Romero Parada y José Alonso Torres Maravilla, por ser de igual forma mi impulso. No abandonarme cuando las cosas se complicaban. Por escucharme, hacerme reír y enseñarme que todo en esta vida es disfrutar con las personas que te hacen sentir completa y llena de vida. Por hacerme sentir importante, acompañada y cuidada. Ser mis cómplices en todo, mis compañeros de trabajo y experiencias locas. Ser mis mejores amigos desde años atrás, demostrándolo cada día de nuestra existencia.

A mis profesores Juan Ramos Ramos, José Federico Ramírez Cruz y Eduardo Sánchez Lucero, por ser mi motivación, pudieron ver mi potencial, plasmaron su total confianza en lo que soy y puedo dar al mundo. Darme siempre ese impulso de esperanza, agradecimiento y orgullo. Buscar hasta por debajo de las piedras las oportunidades que pudieran impulsarme a la cima del éxito.

Finalmente, agradezco a todas aquellas personas que ya no forman parte de mi vida, pero sí de mi alma. Son parte de mi esencia, de lo que soy hoy en día. Con ustedes aprendí todo lo que quería y no quería en mi vida. Gracias por darme de ustedes un pedazo de alma. Por apoyarme en los obstáculos que en su momento atravesaba. Por escucharme y permitirme ser escuchada. En mi corazón habrá siempre un pequeño espacio para cada uno de ustedes.

El futuro me depara grandes experiencias, buenas y malas. Pero hoy sé que lo importante de dar pasos es disfrutar cada uno de ellos, de la mano de todos aquellos que nos aman y aprecian tenernos en nuestras vidas. Gracias vida por este logro y por permitirme ver las grandes maravillas que tienes para ofrecerme a mí y al mundo.

Índice

1. Generalidades del proyecto	11
1.1. Introducción	11
1.2. Descripción de la empresa u organización y del puesto o área del trabajo del estudiante	11
1.3. Problemática	13
1.4. Objetivos	14
1.4.1. Objetivo general	14
1.4.2. Objetivos específicos	14
1.5. Justificación	14
1.6. Delimitación	15
2. Marco teórico	16
2.1. Antecedentes y contexto	16
2.2. Fundamentos teóricos	16
2.2.1. Sistemas de Información y Gestión Operativa	16
2.2.2. Arquitectura Cliente–Servidor	17
2.2.3. Framework Django	18
2.2.4. Diseño Web Responsivo	19
2.2.5. Bases de Datos Relacionales	20
2.2.6. Seguridad y Control de Accesos	20
2.2.7. Pasarela de Pago: PayPal	21
2.2.8. Generación de Reportes en PDF	21
2.2.9. Fundamentos Metodológicos	22
2.2.10. Relación con las Normas y Estándares de Ingeniería	22
2.2.11 Soluciones existentes	22
2.2.12. Diseño de Experiencia de Usuario (UX) e Interfaz de Usuario (UI)	25
Análisis y Diseño del Sistema	27
3.1. Introducción	27
3.2. Análisis del Sistema	27
3.2.1. Descripción general del sistema propuesto	27

3.2.2. Objetivos del análisis del sistema	27
3.2.3. Contexto operativo	28
3.2.4. Beneficios esperados.....	29
3.3. Requerimientos del Sistema	30
3.3.1. Requerimientos funcionales.....	30
3.3.2. Requerimientos no funcionales	32
3.3.3. Requerimientos de hardware y software	35
3.3.4. Conclusión de requerimientos	36
3.4. Modelado de Casos de Uso	37
3.4.1. Actores del sistema	37
3.4.2. Descripción general de los casos de uso	38
3.4.3. Diagrama general de casos de uso	40
3.4.4. Diagramas de casos de uso específicos	41
3.4.5. Descripción de casos de uso principales.....	42
3.5. Modelado del Sistema.....	48
3.5.1. Diagrama de clases.....	49
3.5.2. Modelos	51
3.5.3. Formularios (Forms).....	56
3.5.4. Vistas (Views).....	58
3.5.5. Utilidades y Helpers	59
3.5.6 Diagrama entidad–relación (E–R).....	62
3.5.7. Diagrama de secuencia	14
3.5.8. Diagrama de actividades	21
3.5.9. Diagrama de despliegue.....	22
3.5.10. Conclusión del modelado	23
3.6. Diseño de la Base de Datos	24
3.6.1. Arquitectura general del modelo de datos	26
3.6.2. Descripción detallada de las entidades.....	27
3.6.3. Relaciones entre entidades.....	31

3.6.4. Normalización, integridad y optimización	32
3.6.5. Escalabilidad y modularidad	33
3.6.6. Conclusión del diseño de base de datos	33
3.7. Diseño de la Interfaz.....	33
3.7.1. Panel Administrativo.....	34
3.7.2. Panel del Desarrollador	39
3.7.4. Principios de UX/UI Aplicados	47
3.7.5. Navegación entre Pantallas	49
3.7.6. Conclusión del diseño de interfaz	50
3.8. Conclusión del Diseño.....	50
4.2. Arquitectura del Sistema	51
4.2.1. Estructura General Cliente–Servidor.....	52
4.2.2. Patrón Modelo–Vista–Plantilla (MVT) en Django	52
4.2.3. Diagrama de Arquitectura General	53
4.2.4. Herramientas y Frameworks Utilizados	53
4.3. Estructura del Proyecto.....	54
4.3.1. Organización General de Carpetas	55
4.3.2. Aplicaciones Principales del Sistema	59
4.3.3. Configuración de Rutas y Vistas Principales.....	61
4.3.4. Integración de Plantillas Base y Herencia de Vistas.....	61
4.5. Desarrollo del Panel Administrativo Interno.....	62
4.5.1. Descripción General del Módulo	63
4.5.2. Control de Acceso y Roles de Usuario.....	64
4.5.3. Gestión de Entidades Principales	65
4.5.4. Generación de Reportes en PDF	67
4.5.5. Panel de Métricas e Indicadores	68
4.5.6. Registro de Novedades o Auditorías	69
4.6. Desarrollo del Panel de Desarrollador.....	69
4.6.1. Descripción General y Propósito.....	70

4.6.2. Configuración Institucional del Sistema	71
4.6.3. Gestión Avanzada de Usuarios y Permisos	72
4.6.4. Módulo de Servicios Dinámicos	73
4.6.5. Administración de Medios Estáticos	74
4.6.6. Copias de Seguridad y Mantenimiento	75
4.7. Pruebas y Validación del Sistema	76
4.7.1. Tipos de Pruebas Aplicadas.....	76
4.7.2. Escenarios de Prueba.....	78
4.7.3. Retroalimentación del Usuario o Empresa	80
4.8. Despliegue del Sistema	81
4.8.1. Preparación del Entorno de Producción	82
4.8.2. Implementación en Servidor o Hosting	83
4.8.3. Mantenimiento y Actualizaciones Futuras	85
4.9. Resultados del Desarrollo.....	86
4.9.1. Funcionalidades Implementadas	87
4.9.2. Beneficios Obtenidos	88
4.9.3. Limitaciones Detectadas	89
4.9.4. Valor Agregado al Entorno Empresarial	90
4.10. Conclusiones Parciales.....	91
5.Conclusiones.....	93
5.1. Evaluación de Objetivos	93
5.1.1. Introducción	93
5.1.2. Objetivos del Proyecto	93
5.1.3. Resumen de Cumplimiento.....	96
5.1.4. Observaciones	96
5.1.5. Funcionalidades Adicionales Implementadas	96
5.2. Conclusiones	97
5.2.1. Sobre el Proceso de Desarrollo	97
5.2.2. Sobre los Resultados Técnicos	98

5.2.3. Sobre la Funcionalidad del Sistema	98
5.2.4. Sobre el Impacto y Aplicabilidad	99
5.3. Recomendaciones	99
5.3.1. Recomendaciones de Corto Plazo (1-3 meses).....	99
5.3.2. Recomendaciones de Mediano Plazo (3-6 meses)	100
5.3.3. Recomendaciones de Largo Plazo (6-12 meses)	101
5.3.4. Recomendaciones de Mantenimiento	102
5.4.5. Recomendaciones de Mejora de Experiencia de Usuario	103
5.5. Reflexiones Finales	104
5.5.1. Aspectos destacables del proyecto	104
5.5.2. Lecciones aprendidas	104
Referencias bibliográficas	105
Anexo 1. Manual de Usuario- Panel Administrativo	107
Guía para Administradores.....	107
Guía para Desarrolladores (Panel de Desarrollo)	124
Anexo 2. Manual Técnico	146
1. Introducción	146
1.1 Descripción del Sistema.....	146
1.2 Características Principales	146
1.3 Tecnologías Utilizadas.....	147
2. Arquitectura del Sistema	148
2.1 Arquitectura General	148
2.2 Arquitectura Multi-Tenant	148
2.3 Módulos del Sistema	149
2.4 Flujo de Datos.....	149
3. Requisitos del Sistema	150
3.1 Requisitos de Hardware.....	150
3.2 Requisitos de Software	150
4. Instalación y Configuración.....	152

4.1 Preparación del Entorno	152
4.2 Instalación de Dependencias	152
4.3 Configuración de Base de Datos	153
4.4 Ejecutar Migraciones	154
4.5 Crear Superusuario.....	154
4.6 Crear Empresa Inicial	154
4.7 Cargar Contenido Inicial (Opcional)	155
4.8 Recopilar Archivos Estáticos	155
4.9 Ejecutar Servidor de Desarrollo	156
5. Estructura del Proyecto	157
5.1 Estructura de Directorios	157
5.2 Archivos de Configuración Principales	159
6. Configuración de Base de Datos	160
6.1 Modelos Principales.....	160
6.2 Relaciones Principales	161
6.3 Índices y Optimización	162
7. Módulos Principales	163
7.1 Módulo Home.....	163
7.2 Módulo AdminPanel	163
7.3 Módulo DevPanel	164
7.4 Módulo Empresas	165
7.5 Módulo Servicios	165
7.6 Módulo Reservaciones	166
7.7 Módulo Usuarios	166
8. APIs y Endpoints	167
8.1 URLs Principales.....	167
8.2 Endpoints de E-commerce	168
8.3 Endpoints de Reservaciones.....	168
9. Despliegue en Producción	168

9.1 Arquitectura Recomendada	168
9.2 Configuración con Nginx y Unicorn	169
6. Configurar SSL con Let's Encrypt.....	171
6.1 Configuración de Seguridad en Producción	171
6.2 Configuración de Firewall	172
7. Mantenimiento	173
7.1 RespalDOS de Base de Datos.....	173
7.2 RespalDOS de Archivos Media.....	173
7.3 Actualización del Sistema	173
7.4 Monitoreo de Logs	174
7.5 Optimización de Rendimiento	175
7.6 Limpieza de Datos Temporales.....	176
8. Solución de Problemas.....	177
8.1 Errores Comunes	177
8.2 Debugging	178
9. Seguridad	180
9.1 Buenas Prácticas de Seguridad	180
9.2 Configuración de Seguridad en settings.py	180
9.3 Protección de Datos Sensibles	181
10. Anexos	181
10.1 Comandos de Gestión Útiles	181
10.2 Variables de Entorno Recomendadas	181
10.3 Estructura de Base de Datos	182
10.4 Recursos Adicionales	182
Conclusión	182

1. Generalidades del proyecto

1.1. Introducción

La digitalización de los procesos administrativos internos se ha vuelto un factor determinante para la competitividad de las pequeñas y medianas empresas (PyMEs). No basta con contar con una página web informativa o una plataforma de reservaciones: las empresas requieren herramientas que permitan organizar su operación diaria, controlar sus servicios, llevar un seguimiento de ventas y reservaciones, así como gestionar a su personal y a sus clientes desde un entorno centralizado.

En este contexto se desarrolla el **Sistema Integral de Gestión Operativa Web para PyMEs**, cuyo alcance general se estructura en dos grandes fases:

- Fase 1: Plataforma pública para clientes.
- **Fase 2: Panel administrativo interno.**

El presente documento se enfoca exclusivamente en la **Fase 2**, correspondiente al **panel administrativo interno**, el cual fue diseñado para los administradores, encargados y empleados de las empresas registradas, con el objetivo de ofrecer una herramienta robusta para la gestión de clientes, servicios, productos, reservaciones, ventas, validación de pagos por transferencia, generación de reportes e indicadores, así como la administración básica de usuarios internos.

Este panel, implementado como la aplicación adminpanel/ dentro del proyecto Django, constituye el núcleo operativo del sistema desde la perspectiva de la empresa, ya que concentra la información clave del negocio y permite tomar decisiones basadas en datos actualizados y confiables.

1.2. Descripción de la empresa u organización y del puesto o área del trabajo del estudiante

El proyecto se desarrolla en el Tecnológico Nacional de México, Instituto Tecnológico de Apizaco (TecNM-Apizaco), una institución pública de educación superior con más de 50 años de trayectoria, fundada el 1ro de octubre de 1975, que forma parte del Tecnológico Nacional de México. Ofrece 12 opciones de formación

en ingeniería, 3 maestrías y 1 doctorado, Su objetivo es impulsar el desarrollo profesional y tecnológico en la región. Participa en concursos académicos, culturales y deportivos, fomentando la innovación, la competitividad y el liderazgo, impulsando a los estudiantes a emprender y desarrollar productos innovadores.

La residencia profesional se realiza en el Departamento de Sistemas y Computación, donde el estudiante colabora en el área de desarrollo de software. Las actividades incluyen el análisis de requerimientos, diseño, implementación, pruebas y documentación de los módulos que conforman la plataforma integral. Asimismo, el trabajo involucra el uso de herramientas y frameworks modernos como Django, Bootstrap, y PayPal SDK, aplicando metodologías ágiles y estándares de seguridad web.

Dirección: Av. Instituto Tecnológico No. 418, San Andrés Ahuashuatepec, Municipio de Tzompantepec, Tlaxcala, Méx. C.P. 90491.

Teléfono: 2414172010

Sitio Web: <https://www.apizaco.tecnm.mx/>

Misión: Formar profesionistas que sean líderes visionarios, emprendedores, con capacidad de generar soluciones disruptivas, innovando en la ciencia y la tecnología, comprometidos con el desarrollo económico, responsabilidad social teniendo una visión sostenible y sustentable.

Visión: Ser una Institución de educación superior tecnológica de vanguardia, formadora de ciudadanos del mundo, con conocimientos intercultural, respetando la diversidad social.

Valores:

- **Honestidad:** Se demuestra día a día a través de la transparencia con la ejecución de las actividades laborales cotidianas.
- **Responsabilidad:** Es la actitud que caracteriza al personal en el cumplimiento de su tarea con eficacia y eficiencia en beneficio de la comunidad.

- **Lealtad:** Es la actitud que caracteriza al personal del Instituto Tecnológico de Apizaco, aplica sus conocimientos y habilidades en el logro de las metas institucionales, sin olvidar en ningún momento el lema: “Pensar para servir, servir para triunfar ®”.

Giro de la empresa: Publica y de servicios.

1.3. Problemática

Antes de la implementación de este panel, las PyMEs del sector turismo que se buscaba apoyar presentaban problemas como:

- **Procesos administrativos fragmentados**, donde las reservaciones en línea no se relacionaban automáticamente con el control interno de ventas o el seguimiento de clientes.
- **Uso de herramientas manuales** (hojas de cálculo, cuadernos de notas, mensajería instantánea) para registrar reservaciones, pagos y estados, generando duplicidad de información y errores.

proyecto final

- **Ausencia de un control de accesos por roles**, lo que hacía difícil limitar quién podía modificar información sensible (por ejemplo, cambios en precios, estados de reservaciones o datos de clientes).
- **Falta de reportes consolidados** en un formato estándar (como PDF) para apoyar la toma de decisiones y el análisis de ventas y reservaciones.
- **Carencia de trazabilidad** de acciones internas, de modo que no era claro qué usuario había validado un pago, modificado un registro o cancelado una reservación.

Estos problemas se traducían en **pérdida de tiempo**, posibilidad de errores humanos, dificultades para tomar decisiones informadas y una baja capacidad para auditar o revisar la operación interna.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar e implementar un **panel administrativo interno web**, integrado al Sistema Integral de Gestión Operativa, que permita a PyMEs del sector servicios y turismo gestionar de forma centralizada sus clientes, empleados, servicios, reservaciones, ventas y pagos, con control de accesos por roles, generación de reportes e indicadores, garantizando la seguridad, trazabilidad y escalabilidad del sistema.

1.4.2. Objetivos específicos

- Diseñar una interfaz interna segmentada por módulos (clientes, empleados, servicios, reservaciones, ventas) orientada a la operatividad diaria de la empresa.
- Implementar un control de acceso basado en roles (administrador, empleado, encargado) que limite y adapte las funcionalidades visibles de acuerdo con el perfil del usuario.
- Integrar la gestión interna de reservaciones, vinculada a la información proveniente de la plataforma pública y al estado de los pagos (PayPal y transferencia bancaria).
- Desarrollar módulos de validación de pagos por transferencia, permitiendo revisar comprobantes y cambiar estados de reservación de forma controlada.
- Incorporar reportes en formato PDF e indicadores visuales (panel de métricas) que muestren ventas, reservaciones y otros datos relevantes.
- Registrar las acciones internas en un módulo de novedades o auditorías, para asegurar la trazabilidad y transparencia de la operación.

1.5. Justificación

El panel administrativo interno es la pieza que transforma la plataforma en una **solución integral de gestión**, ya que no solo permite atender al cliente final, sino también organizar la operación interna de la empresa. Entre las principales razones para su desarrollo se encuentran:

- Responder a la necesidad de **centralizar la información** de reservaciones, ventas y clientes, evitando la fragmentación de datos entre distintos sistemas o archivos.
- Proporcionar una herramienta que **automatice tareas operativas** y reduzca los errores asociados a registros manuales (por ejemplo, duplicidad de reservaciones, errores en montos o fechas).
- Implementar **medidas de seguridad y control de accesos** que garanticen la confidencialidad de la información y eviten modificaciones no autorizadas.
- Facilitar la toma de decisiones mediante la disponibilidad de reportes y métricas actualizadas sobre la operación de la empresa.
- Aportar un **componente reutilizable y escalable**, útil para otras empresas del sector que requieran un sistema de administración interna similar.

1.6. Delimitación

Este documento se limita exclusivamente a la **Fase 2: Panel administrativo interno**. No se aborda en detalle la plataforma pública (Fase 1) ni el panel de desarrollador, salvo en los puntos donde se hace necesario explicar interacciones entre módulos (por ejemplo, el impacto de los pagos realizados en el sitio público sobre la gestión interna de reservaciones).

El panel administrativo cubre:

- Gestión interna de clientes, empleados, servicios, productos, reservaciones y ventas.
- Validación de pagos por transferencia bancaria.
- Generación de reportes en PDF, panel de métricas y registro de novedades o auditorías.

No se incluyen en esta fase:

- Integraciones externas con sistemas de contabilidad o facturación electrónica.
- Módulos avanzados de configuración de temas y servicios dinámicos (propios del panel de desarrollador).

2.Marco teórico

2.1. Antecedentes y contexto

Las PyMEs representan más del 99% de las unidades económicas en México y generan alrededor del 70% del empleo formal; no obstante, enfrentan serias limitaciones tecnológicas, especialmente en lo referente a su **presencia digital y gestión de la relación con clientes**.

En el caso específico del turismo, los usuarios esperan:

- Consultar servicios y precios antes de visitar el lugar.
- Reservar en línea desde su dispositivo móvil.
- Pagar de forma segura sin necesidad de trasladarse físicamente.

Esto hace que una **plataforma web pública especializada en servicios y reservaciones turísticas** sea una pieza clave en la modernización y competitividad de las empresas del sector.

2.2. Fundamentos teóricos

2.2.1.Sistemas de Información y Gestión Operativa

Un sistema de información (SI) se define como el conjunto de componentes interrelacionados que recolectan, procesan, almacenan y distribuyen información con el propósito de apoyar la toma de decisiones, la coordinación y el control dentro de una organización (Laudon & Laudon, 2016). Estos sistemas constituyen el núcleo informático de las organizaciones modernas, al permitir la automatización de tareas, la integración de áreas funcionales y la optimización de procesos estratégicos.

Dentro de las PyMEs, los sistemas de gestión operativa web representan una herramienta fundamental para mejorar la eficiencia, ya que permiten centralizar y estandarizar la información proveniente de distintas áreas como ventas, inventarios, reservas o atención al cliente.

La incorporación de tecnologías de Internet en estos sistemas transforma el modelo operativo tradicional hacia una gestión digital en tiempo real, lo que facilita la supervisión de las operaciones desde cualquier ubicación y dispositivo.

La gestión operativa web también impulsa la toma de decisiones basada en datos, ya que los sistemas pueden generar reportes, métricas y análisis automáticos que reflejan el rendimiento empresarial. De esta forma, las empresas no solo optimizan sus recursos humanos y financieros, sino que logran mejorar la satisfacción del cliente, al proporcionar experiencias digitales más rápidas, seguras y personalizadas.

2.2.2.Arquitectura Cliente–Servidor

El modelo cliente–servidor constituye la base estructural del proyecto, dado que permite dividir las responsabilidades entre el cliente (interfaz de usuario) y el servidor (procesamiento de datos y lógica de negocio). En este modelo, el cliente envía peticiones mediante un navegador web, mientras que el servidor procesa dichas solicitudes, accede a la base de datos y devuelve las respuestas en formato HTML, JSON o XML, según el caso.

Este paradigma es ampliamente utilizado en el desarrollo de aplicaciones web modernas debido a sus ventajas de escalabilidad, modularidad y seguridad. El servidor puede gestionar múltiples conexiones simultáneamente y mantener sesiones activas con diferentes usuarios. En el caso del Sistema Integral de Gestión Operativa Web, el cliente está representado por las interfaces diseñadas con HTML, CSS y Bootstrap, mientras que el servidor está implementado en Django, el cual gestiona las peticiones, las operaciones sobre la base de datos y la autenticación de usuarios.

La arquitectura cliente–servidor también se complementa con el uso del protocolo HTTP/HTTPS, que permite la comunicación segura entre ambos extremos, evitando la manipulación de datos durante la transmisión.

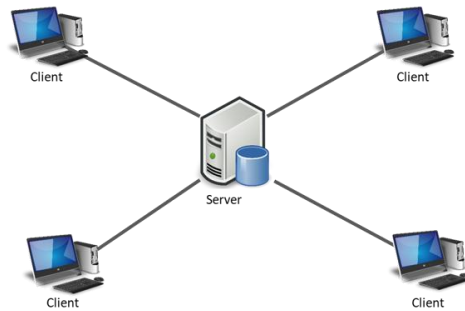


Figura 1: Arquitectura Cliente–Servidor

2.2.3. Framework Django

Django es un framework de desarrollo web de código abierto basado en el lenguaje Python, que sigue el patrón arquitectónico Modelo–Vista–Plantilla (MVT). Este modelo promueve una separación clara entre la lógica de negocio, la presentación y la persistencia de datos, lo que facilita el mantenimiento, la escalabilidad y la reutilización del código.

- **Modelo (Model):** define las estructuras de datos mediante clases que representan tablas en la base de datos. Django utiliza un Object-Relational Mapping (ORM) que traduce automáticamente las operaciones de Python en consultas SQL, eliminando la necesidad de escribir sentencias manuales.
- **Vista (View):** controla la lógica del sistema, gestiona las solicitudes del usuario y prepara los datos que serán enviados a las plantillas.
- **Plantilla (Template):** define la estructura visual de las páginas, utilizando un motor de plantillas que permite combinar código HTML con etiquetas de Django para mostrar los datos dinámicamente.

Django ofrece ventajas significativas, como un panel administrativo autogenerado, autenticación de usuarios, manejo integrado de sesiones y protección contra vulnerabilidades como la inyección SQL, el Cross-Site Scripting (XSS) y el Cross-Site Request Forgery (CSRF). Estas características lo convierten en una herramienta ideal para la construcción de sistemas empresariales robustos, seguros y modulares.



Figura 2: Framework Django

2.2.4. Diseño Web Responsivo

El diseño web responsivo (Responsive Web Design, RWD) es un enfoque de desarrollo que permite que las interfaces se adapten de manera automática al tamaño y resolución de cualquier dispositivo, ya sea computadora, tableta o teléfono móvil.

Este principio es esencial en el contexto actual, donde el acceso a los servicios digitales se realiza desde una amplia gama de dispositivos.

Para la implementación del diseño responsivo, se emplea el framework Bootstrap, que utiliza un sistema de cuadrículas (*grid system*), clases CSS predefinidas y componentes visuales reutilizables. De esta manera, el sistema garantiza una experiencia de usuario uniforme, independientemente del dispositivo desde el cual se accede.

El diseño responsivo no solo mejora la usabilidad, sino que también contribuye al posicionamiento SEO y a la accesibilidad, dos factores importantes para la visibilidad y la competitividad de las empresas en entornos digitales.



Figura 3: Bootstrap

2.2.5. Bases de Datos Relacionales

Las bases de datos relacionales (BDR) son estructuras lógicas que almacenan información en tablas compuestas por filas y columnas. Cada tabla representa una entidad del sistema (por ejemplo, usuarios, reservaciones o productos), y las relaciones entre ellas se definen mediante llaves primarias y foráneas.

El modelo relacional, propuesto por Edgar F. Codd en 1970, garantiza consistencia, integridad y seguridad en los datos mediante operaciones algebraicas conocidas como consultas SQL.

En este proyecto, se utilizó una base de datos relacional compatible con Django (MySQL o PostgreSQL), administrada a través de su ORM. Este enfoque permite abstraer la complejidad del SQL, facilitando operaciones como filtrado, inserción o actualización de datos con instrucciones de alto nivel en Python.

Gracias a esta integración, el sistema mantiene una estructura de datos coherente, segura y escalable, lo que resulta indispensable para la gestión de información crítica como pagos, reservaciones y registros de clientes.



Figura 4: SQL

2.2.6. Seguridad y Control de Accesos

Aunque este documento se centra en la plataforma pública, la seguridad es crítica, ya que se manejan **datos personales y de pago**. Entre las medidas implementadas se encuentran:

- Cifrado de contraseñas mediante algoritmos hash.
- Uso de **HTTPS** para proteger la transmisión de datos.
- Protección CSRF en formularios.
- Validaciones en servidor y cliente.

2.2.7. Pasarela de Pago: PayPal

La integración de PayPal proporciona una solución de comercio electrónico segura y reconocida internacionalmente, basada en API REST que permite realizar cobros en línea de forma rápida y confiable. PayPal utiliza protocolos de cifrado SSL y autenticación OAuth 2.0, garantizando la protección de las transacciones y la identidad de los usuarios.

Dentro del sistema, PayPal permite que los clientes efectúen pagos directos desde la plataforma, con registro automático del resultado de la transacción (éxito o fallo). Este proceso mejora la experiencia del usuario, reduce tiempos de confirmación y automatiza la conciliación contable dentro del panel administrativo.

De acuerdo con la documentación oficial de PayPal (2024), la API facilita la integración con aplicaciones web basadas en frameworks como Django o Node.js, lo que respalda su elección en este proyecto.



Figura 5: PayPal

2.2.8. Generación de Reportes en PDF

Los reportes administrativos son herramientas indispensables para el seguimiento de resultados y la toma de decisiones. El sistema implementa módulos que generan reportes en formato PDF, utilizando librerías como reportlab o xhtml2pdf, las cuales permiten transformar información dinámica (consultas a la base de datos) en documentos formateados.

Estos reportes incluyen datos relevantes como ventas, reservaciones, servicios y empleados, proporcionando una visión analítica del rendimiento empresarial. Además, el formato PDF asegura la portabilidad, compatibilidad y protección del contenido, permitiendo su almacenamiento o impresión en cualquier entorno operativo.

2.2.9. Fundamentos Metodológicos

El desarrollo del sistema se basó en la metodología **SCRUM**, que permite gestionar proyectos de software de forma ágil, mediante iteraciones cortas llamadas *sprints*. Cada sprint contempló actividades específicas como:

- **Análisis de requerimientos.** Revisión de las necesidades funcionales y no funcionales de las empresas participantes.
- **Diseño y modelado.** Elaboración de diagramas UML y estructura de base de datos.
- **Desarrollo e integración.** Construcción progresiva de módulos funcionales para cada fase.
- **Pruebas.** Validación de la funcionalidad, seguridad y rendimiento.
- **Entrega y retroalimentación.** Evaluación final con los asesores y usuarios.

Esta metodología garantizó un desarrollo **colaborativo, iterativo y controlado**, permitiendo realizar ajustes durante el proceso y asegurar la calidad del producto final.

2.2.10. Relación con las Normas y Estándares de Ingeniería

El proyecto se alinea con las buenas prácticas de la ingeniería de software descritas por el **IEEE (Institute of Electrical and Electronics Engineers)**, especialmente las relacionadas con el proceso de desarrollo, documentación, pruebas y mantenimiento de sistemas. Asimismo, cumple con los principios de **usabilidad y accesibilidad** definidos por la **W3C (World Wide Web Consortium)**, garantizando que la plataforma sea intuitiva, funcional y accesible para todo tipo de usuarios.

2.2.11 Soluciones existentes

Para comprender la pertinencia del "Sistema Integral de Gestión Operativa Web", se realizó un análisis de las soluciones tecnológicas actualmente disponibles para las Pequeñas y Medianas Empresas (PyMEs), especialmente en el sector de

servicios y turismo. Estas soluciones se pueden agrupar en las siguientes categorías:

2.2.11.1. Herramientas Manuales y de Ofimática (Gestión no Digitalizada)

Una gran parte de las PyMEs, particularmente las de reciente creación o con menor madurez digital, basan su gestión en herramientas de ofimática como Microsoft Excel para inventarios y finanzas, y servicios de mensajería (como WhatsApp Business) para la atención al cliente y reservaciones.

Esta dependencia de procesos manuales no es por falta de voluntad, sino que a menudo se debe a barreras de adopción significativas. Estudios sobre la digitalización de PyMEs en Latinoamérica identifican que la falta de recursos financieros para la inversión inicial, la percepción de alta complejidad de las tecnologías y la falta de personal con conocimientos técnicos especializados son los principales frenos para la adopción tecnológica (Nurqamarani et al., 2021). El resultado es una gestión propensa a errores humanos, sin información en tiempo real y con "islas de información" (datos fragmentados) que impiden una visión global del negocio.

2.2.11.2. Sistemas ERP (Planificación de Recursos Empresariales) de Gran Escala

En el extremo opuesto del espectro, existen soluciones ERP robustas como SAP Business One, Oracle NetSuite u Odoo. Estos sistemas son monolíticos y ofrecen una gestión integral de todas las áreas de una empresa (finanzas, logística, recursos humanos, etc.).



Figura 6: ODOO

Sin embargo, su principal barrera para la PyME es el costo prohibitivo y la complejidad de implementación. Estos sistemas están diseñados para estandarizar procesos de grandes corporaciones, no para la agilidad de una pequeña empresa. Para una PyME del sector servicios, la implementación de un ERP de esta magnitud a menudo resulta en "sobreingeniería", donde se paga por un exceso de funcionalidades que no se requieren, presentando una curva de aprendizaje demasiado elevada para el personal (Lufti et al., 2022).

2.2.11.3. Fragmentación de Soluciones SaaS (Software como Servicio)

En la última década, ha surgido una explosión de soluciones de Software como Servicio (SaaS) que resuelven problemas específicos de forma muy eficiente. Por ejemplo, una empresa puede usar WordPress para su página web, Mailchimp para correos, un sistema POS (Punto de Venta) para el cobro y una plataforma como Booking.com para las reservaciones.

Aunque efectivas en su nicho, esta estrategia conduce a un fenómeno conocido como "SaaS Sprawl" o proliferación de aplicaciones (IBM, 2023). La empresa termina pagando múltiples suscripciones mensuales y su información crítica queda secuestrada en silos de datos. El personal debe transferir información manualmente entre sistemas (ej. pasar una reserva de Booking al Excel de contabilidad), creando redundancia, inconsistencias y una falta total de visibilidad operativa unificada.

2.2.11.4. Identificación de la Oportunidad (El "Gap")

El análisis del estado del arte revela una brecha significativa en el mercado: las PyMEs requieren una solución que se encuentre en el punto medio. Necesitan un sistema que supere las limitaciones de las herramientas manuales, pero que no incurra en la complejidad y el costo de un ERP de gran escala.

El "Sistema Integral de Gestión Operativa Web" propuesto se posiciona en ese vacío. Busca ofrecer las ventajas de la centralización y la digitalización, pero con un enfoque modular, accesible y adaptado a las necesidades específicas de las PyMEs (gestión de reservaciones, catálogo de servicios y panel de cliente), integrando la plataforma pública (Fase 1) y el panel administrativo (Fase 2) en una solución coherente y escalable.

2.2.12. Diseño de Experiencia de Usuario (UX) e Interfaz de Usuario (UI)

El diseño de Experiencia de Usuario (User Experience, UX) y el diseño de Interfaz de Usuario (User Interface, UI) son componentes esenciales en el desarrollo de sistemas web modernos. Su objetivo principal es garantizar que las aplicaciones no solo funcionen correctamente, sino que también resulten intuitivas, accesibles y satisfactorias para los usuarios finales.

De acuerdo con Norman (2013), la experiencia de usuario abarca todos los aspectos de la interacción del usuario con un producto, sistema o servicio. No se limita únicamente a la estética, sino que incluye factores como la facilidad de uso, la eficiencia, la percepción emocional y la coherencia del flujo de navegación. En este sentido, un sistema con buena UX permite al usuario cumplir sus objetivos de manera rápida, clara y sin frustración.

Por su parte, la interfaz de usuario (UI) se refiere al conjunto de elementos visuales y de interacción con los que el usuario entra en contacto: botones, menús, formularios, tipografía, iconografía y colores. Una UI efectiva debe ser consistente, atractiva y funcional, guiando al usuario de forma natural a través de las tareas del sistema. Según Garrett (2011), el diseño de UI constituye la capa más visible de la

experiencia digital, pero debe sustentarse sobre una arquitectura de información y una estructura de interacción bien planificadas.

En el contexto del presente proyecto, la fase pública del sistema —orientada a los clientes— y el panel administrativo —destinado a los empleados y administradores— fueron desarrollados aplicando principios de diseño centrado en el usuario (User-Centered Design, UCD). Este enfoque promueve la participación del usuario en todas las etapas del desarrollo, desde el análisis de requerimientos hasta la validación de prototipos, garantizando que las funcionalidades respondan a necesidades reales.

Durante el desarrollo se aplicaron los siguientes principios de UX/UI:

- **Consistencia visual:** uso de estilos homogéneos, tipografía uniforme y paleta de colores coherente con la identidad del sistema.
- **Jerarquía visual:** organización clara de los elementos para destacar las acciones principales (por ejemplo, botones de “Reservar” o “Guardar”).
- **Retroalimentación inmediata:** mensajes y alertas que confirman las acciones del usuario, como el envío de formularios o el registro exitoso.
- **Accesibilidad:** contraste adecuado de colores, textos legibles y navegación optimizada para diferentes dispositivos (mobile-first design).
- **Prevención de errores:** validaciones en formularios y confirmaciones antes de realizar acciones críticas (como eliminar registros).
- **Simplicidad:** minimizar la carga cognitiva mediante interfaces limpias, iconografía clara y procesos guiados paso a paso.

El resultado es una plataforma intuitiva, moderna y coherente, que mejora la interacción tanto del cliente como del personal administrativo. Esto no solo incrementa la eficiencia operativa, sino que también contribuye a la satisfacción del usuario final, un factor clave para el éxito y la adopción de cualquier sistema web.

Análisis y Diseño del Sistema

3.1. Introducción

En este capítulo se presenta el análisis y diseño específico del **panel administrativo interno**, considerando los actores internos, requerimientos funcionales y no funcionales asociados, así como los modelos de casos de uso, clases y entidades que dan soporte a la operación administrativa.

Si bien el sistema completo contempla otros módulos, aquí se restringe el análisis a los elementos que intervienen directamente en la **gestión interna de la empresa**: clientes, empleados, servicios, productos, reservaciones, ventas, pagos y reportes.

3.2. Análisis del Sistema

3.2.1. Descripción general del sistema propuesto

El **panel administrativo interno** es un módulo web restringido a usuarios internos de la empresa (administrador, empleado, encargado). Su propósito es ofrecer una interfaz unificada para la gestión de:

- Clientes y su historial de reservaciones.
- Empleados y roles internos.
- Servicios y productos ofrecidos por la empresa.
- Reservaciones generadas desde el sitio público.
- Ventas y estados de pago.
- Validación de comprobantes de transferencia.
- Reportes en PDF y métricas.
- Registro de novedades o auditorías.
- proyecto final
- El panel se estructura mediante una **barra lateral** con módulos organizados y un **dashboard** inicial que resume la información clave.

Figura 5. Esquema general del panel administrativo interno.

3.2.2. Objetivos del análisis del sistema

- Identificar las entidades y procesos que son administrados por usuarios internos.

- Determinar los requerimientos de seguridad y control de acceso por rol dentro del panel.
- Definir los flujos de información entre la plataforma pública y el panel (reservaciones, pagos).
- Establecer las bases para el diseño de la base de datos y las vistas del panel.

3.2.2.1. Actores y alcance del sistema

Los actores directamente relacionados con el panel administrativo son:

proyecto final

- **Administrador:** tiene acceso completo al panel, puede gestionar clientes, empleados, servicios, reservaciones, ventas, reportes y métricas.
- **Encargado:** tiene mayores permisos operativos que un empleado (por ejemplo, validar pagos y modificar estados de reservaciones), pero no accede a todas las configuraciones administrativas.

proyecto final

- **Empleado:** usuario operativo que puede registrar reservaciones, atender clientes, revisar información y subir comprobantes, pero con permisos limitados.

proyecto final

El alcance del módulo incluye todas las operaciones internas vinculadas con la empresa activa; no administra la configuración global del sistema (que corresponde al panel de desarrollador).

.

3.2.3. Contexto operativo

El desarrollo del sistema se enmarca en el contexto de digitalización de las PyMEs, que en su mayoría carecen de infraestructura tecnológica avanzada. Se optó por una solución 100% web, hospedada en un servidor con soporte para Django, lo que elimina la necesidad de instalación local y permite acceder al sistema desde cualquier dispositivo conectado a internet.

A nivel técnico, el entorno de desarrollo se construyó sobre Python 3.12 y Django 5, con base de datos MySQL o PostgreSQL, gestionada mediante el ORM de Django.

El frontend fue implementado con HTML5, CSS3, Bootstrap 5 y JavaScript, garantizando una experiencia visual moderna y fluida. El uso de herramientas de control de versiones como Git y GitHub permitió llevar un seguimiento preciso del progreso y versiones del sistema durante su desarrollo.

3.2.4. Beneficios esperados

La implementación del sistema Xoletongo conlleva una serie de beneficios tanto técnicos como operativos:

- **Centralización de datos y procesos** en un entorno unificado y accesible.
- **Optimización del tiempo de gestión** de servicios, clientes y reservaciones.
- **Reducción de errores humanos** mediante la automatización de tareas repetitivas.
- **Disponibilidad permanente (24/7)** gracias a su naturaleza web.
- **Mejora en la atención al cliente**, ofreciendo un canal digital eficiente y accesible.
- **Control jerárquico mediante roles**, asegurando que cada usuario tenga acceso solo a las funciones pertinentes.
- **Transparencia administrativa** mediante el registro de acciones (bitácora).
- **Seguridad avanzada**, con cifrado de contraseñas, control CSRF y validación de formularios.
- **Escalabilidad y personalización**, facilitando su adaptación a otras empresas o sectores.
- **Aporte al proceso de transformación digital local**, impulsando la competitividad de las PyMEs.

3.3. Requerimientos del Sistema

El análisis de requerimientos constituye una de las fases más importantes en el ciclo de desarrollo del sistema, ya que permite definir con precisión qué debe hacer el sistema (requerimientos funcionales) y cómo debe hacerlo (requerimientos no funcionales).

Esta etapa se llevó a cabo mediante la recopilación de información, entrevistas con el usuario final, observación directa de procesos empresariales y revisión de herramientas tecnológicas existentes, con el fin de garantizar que la solución propuesta respondiera adecuadamente a las necesidades operativas de las pequeñas y medianas empresas.

El sistema está orientado a la gestión centralizada de servicios, clientes, reservaciones y ventas, así como a la automatización de tareas administrativas y a la mejora de la experiencia del usuario mediante interfaces web intuitivas y seguras.

Los requerimientos aquí descritos fueron clasificados en tres categorías: funcionales, no funcionales y técnicos.

3.3.1. Requerimientos funcionales

A partir de la tabla general de requerimientos del sistema, se destacan para el panel administrativo:

Tabla 1 Requerimientos funcionales del sistema

Código	Requerimiento funcional	Descripción
RF-01	Registro y autenticación de usuarios	El sistema debe permitir el registro, inicio y cierre de sesión para clientes, empleados y administradores.
RF-02	Recuperación de contraseña	El usuario debe poder restablecer su

		contraseña mediante correo electrónico seguro.
RF-03	Gestión de clientes	El administrador y los empleados podrán registrar, consultar, editar y eliminar información de clientes.
RF-04	Gestión de servicios	El administrador podrá crear, editar y eliminar servicios, incluyendo nombre, descripción, costo e imagen.
RF-05	Gestión de ventas y reservaciones	El sistema permitirá registrar y consultar las reservaciones y ventas generadas por los clientes.
RF-06	Control de roles y permisos	El sistema deberá asignar permisos de acceso según el tipo de usuario (administrador, empleado, cliente).
RF-07	Generación de reportes en PDF	El administrador podrá generar reportes administrativos (ventas, clientes, servicios) en formato PDF.
RF-08	Panel de métricas e indicadores	El sistema deberá mostrar estadísticas

		resumidas sobre ventas, usuarios y reservaciones.
RF-09	Configuración institucional	El administrador o desarrollador podrá modificar datos de la empresa (nombre, logo, colores, políticas).
RF-10	Gestión de usuarios internos	El administrador podrá crear cuentas para empleados y asignarles roles específicos.
RF-11	Gestión de novedades o bitácora	El sistema registrará automáticamente las acciones importantes de los usuarios internos.
RF-12	Exportación de datos	El sistema deberá permitir exportar información a formato CSV para respaldo o análisis.
RF-13	Copias de seguridad	El desarrollador podrá realizar respaldos de la base de datos o archivos estáticos del sistema.

3.3.2. Requerimientos no funcionales

Los requerimientos no funcionales establecen las condiciones de calidad, rendimiento y seguridad que debe cumplir el sistema para garantizar su correcto funcionamiento en un entorno real. Estos requerimientos aseguran que la aplicación no solo cumpla con sus funciones, sino que también sea eficiente, confiable, accesible y segura.

Tabla 2 Requerimientos no funcionales del sistema

Código	Requerimiento funcional	Descripción
RNF-01	Usabilidad	El sistema deberá ofrecer una interfaz clara, coherente y fácil de utilizar, aplicando principios UX/UI.
RNF-02	Accesibilidad	La plataforma deberá adaptarse a usuarios con distintos dispositivos y niveles de experiencia.
RNF-03	Diseño responsivo	La interfaz se ajustará automáticamente a distintos tamaños de pantalla (PC, tablet, móvil).
RNF-04	Seguridad	Se emplearán protocolos HTTPS, validación de formularios y cifrado de contraseñas mediante algoritmos hash.
RNF-05	Autenticación y control de acceso	Se implementará un sistema de roles con permisos diferenciados (cliente, empleado, administrador, desarrollador).
RNF-06	Rendimiento	Las páginas deberán cargar en menos de tres segundos bajo

		condiciones normales de conexión.
RNF-07	Escalabilidad	El sistema deberá poder ampliarse para soportar un mayor número de usuarios o módulos.
RNF-08	Mantenibilidad	El código deberá estar estructurado modularmente para facilitar la actualización y corrección de errores.
RNF-09	Compatibilidad	El sistema deberá funcionar correctamente en navegadores modernos (Chrome, Firefox, Edge, Safari).
RNF-10	Integridad de datos	Se garantizará la consistencia de los datos mediante restricciones de base de datos y control de transacciones.
RNF-11	Portabilidad	El sistema podrá ser implementado en distintos servidores web compatibles con Python y Django.
RNF-12	Disponibilidad	El servicio deberá estar disponible las 24 horas del día con una tasa de operación superior al 95%.

RNF-13	Recuperación ante fallos	El sistema deberá contar con mecanismos de respaldo y restauración de datos en caso de error o pérdida.
RNF-14	Privacidad	La información personal de los usuarios será tratada conforme a la Ley Federal de Protección de Datos Personales.
RNF-15	Documentación	Todo el sistema deberá contar con documentación técnica y manual de usuario para facilitar su mantenimiento.

3.3.3. Requerimientos de hardware y software

Esta sección especifica los recursos necesarios tanto para el entorno del cliente (usuario final) como del servidor (plataforma de alojamiento). La configuración propuesta garantiza un desempeño óptimo y estabilidad en la ejecución del sistema.

Tabla 3 Requerimientos de hardware y software

Componente	Mínimo requerido	Recomendado	Descripción / uso
Servidor web	Procesador Intel Core i3, 4 GB RAM, 40 GB disco	Procesador Intel Core i5/i7, 8 GB RAM, 80 GB disco	Servidor que aloja el sistema Django y la base de datos.
Sistema operativo servidor	Ubuntu 20.04 LTS o Windows Server 2019	Ubuntu 22.04 LTS	Sistema operativo del servidor de despliegue.

Software servidor	Python 3.10+, Django 5.0+, MySQL o PostgreSQL, Apache/Nginx	Python 3.12, Django 5.0, PostgreSQL, Unicorn + Nginx	Entorno de ejecución y dependencias principales.
Cliente (usuario)	Navegador web moderno (Chrome, Firefox, Edge)	Última versión de Chrome o Edge	Navegador para acceder al sistema web.
Sistema operativo cliente	Windows 10, Android 10 o iOS 13	Windows 11, Android 14 o iOS 17	Dispositivo del usuario para acceso web.
Herramientas de desarrollo	Visual Studio Code, Git, Lucidchart o Draw.io	Cursor, GitHub Desktop, Figma, StarUML	Utilizadas para el desarrollo, modelado y control de versiones.
Conectividad	Conexión a Internet estable (1 Mbps mínimo)	10 Mbps o superior	Requisito para el correcto funcionamiento del sistema.

3.3.4. Conclusión de requerimientos

El conjunto de requerimientos definidos en esta etapa permitió establecer los criterios técnicos y funcionales que guiarán la fase de diseño y desarrollo del sistema.

Estos requerimientos aseguran que la solución sea coherente, segura, flexible y alineada con las necesidades reales de las PyMEs, garantizando que cada módulo cumpla con un propósito específico dentro del flujo general de gestión operativa.

3.4. Modelado de Casos de Uso

El modelado de casos de uso constituye una de las herramientas más importantes dentro del análisis orientado a objetos, ya que permite representar de manera gráfica y descriptiva la interacción entre los usuarios (actores) y el sistema. A través de este modelo se identifican las funcionalidades principales que el sistema debe ofrecer y los diferentes escenarios que se presentan durante su uso.

El objetivo de esta sección es definir los casos de uso del Sistema Integral de Gestión Operativa Web para PyMEs, estableciendo con claridad quiénes intervienen, qué acciones realizan y qué resultados se esperan. Estos modelos fueron elaborados utilizando el lenguaje UML (Unified Modeling Language) y se generaron mediante herramientas de modelado como Lucidchart y Draw.io.

3.4.1. Actores del sistema

En el sistema se identificaron tres actores principales, cada uno con un conjunto de responsabilidades y privilegios específicos:

Tabla 4 Actores del sistema

Actor	Descripción	Actor
Empleado	Usuario interno con permisos limitados. Puede gestionar reservaciones y atender solicitudes de los clientes, sin acceso a configuraciones globales.	Empleado
Administrador	Usuario con control total sobre el sistema. Supervisa la gestión de servicios, clientes,	Administrador

	usuarios internos y reportes administrativos.	
Desarrollador	Actor técnico responsable del mantenimiento del sistema. Tiene acceso a configuraciones avanzadas, respaldos, exportaciones y personalización de la plataforma.	Desarrollador

Cada uno de estos actores participa en uno o varios casos de uso, de acuerdo con su nivel de interacción y permisos dentro del sistema.

3.4.2. Descripción general de los casos de uso

Los casos de uso representan de manera estructurada los diferentes escenarios de interacción entre los actores y el sistema, especificando los objetivos que cada usuario busca alcanzar al utilizar la aplicación. Cada caso de uso describe una secuencia de pasos o eventos que conducen al cumplimiento de una tarea específica dentro del sistema, asegurando que las funciones implementadas respondan directamente a las necesidades reales de los usuarios finales.

En este contexto, los casos de uso permiten identificar no solo las funcionalidades principales del sistema, sino también la relación entre los distintos módulos, las condiciones previas y los resultados esperados tras la ejecución de cada proceso. De esta forma, se establece una comunicación clara entre los requerimientos funcionales definidos anteriormente y su futura implementación en el diseño y desarrollo del software.

3.4.2.1. Casos de Uso: Empleado

El segundo grupo de casos de uso involucra al empleado o usuario interno con permisos limitados. Su papel dentro del sistema consiste en gestionar la atención al

cliente, registrar nuevas reservaciones, verificar pagos realizados, y actualizar el estado de los servicios o ventas. Este actor no tiene acceso a configuraciones avanzadas ni a reportes financieros, pero su intervención es clave para mantener actualizado el flujo operativo diario. El sistema le permite realizar estas tareas de forma controlada y supervisada, evitando conflictos de permisos y garantizando la integridad de los datos.

3.4.2.2. Casos de Uso: Administrador

El tercer grupo de casos de uso pertenece al administrador, quien posee control total sobre la plataforma. Sus acciones incluyen gestionar usuarios internos, registrar nuevos servicios, generar reportes administrativos, supervisar métricas del sistema y configurar parámetros institucionales como el nombre de la empresa, logotipo, colores corporativos o políticas de privacidad. Además, el administrador puede revisar las bitácoras de actividad interna, lo que permite mantener un registro transparente de las operaciones realizadas por el personal. Este conjunto de casos de uso representa el núcleo de la administración del sistema y garantiza el control y la trazabilidad de las acciones realizadas por los distintos actores.

3.4.2.3. Casos de Uso: Desarrollador

Por último, se encuentra el actor desarrollador, responsable de la gestión técnica y el mantenimiento del sistema. Sus casos de uso incluyen la exportación de información en formato CSV, la realización de respaldos de la base de datos, la personalización del entorno institucional, la creación de nuevos roles y la supervisión de las novedades registradas por el sistema. Este actor se considera de nivel avanzado y su intervención está enfocada en la sostenibilidad técnica del sistema, asegurando su correcta operación, actualización y seguridad a largo plazo.

En conjunto, los casos de uso definen una visión completa del comportamiento esperado del sistema desde la perspectiva del usuario. Permiten entender cómo cada módulo contribuye al cumplimiento de los objetivos generales y cómo las diferentes funciones se integran de manera coherente. Gracias a esta estructuración, se facilita el diseño posterior de los diagramas UML, que representan

gráficamente las interacciones descritas y sirven como referencia directa durante la fase de implementación.

3.4.3. Diagrama general de casos de uso

El Diagrama General de Casos de Uso muestra una vista global de la interacción entre los actores y las principales funcionalidades del sistema. Este diagrama permite identificar rápidamente qué usuarios participan en cada proceso y cuáles son las dependencias entre ellos.

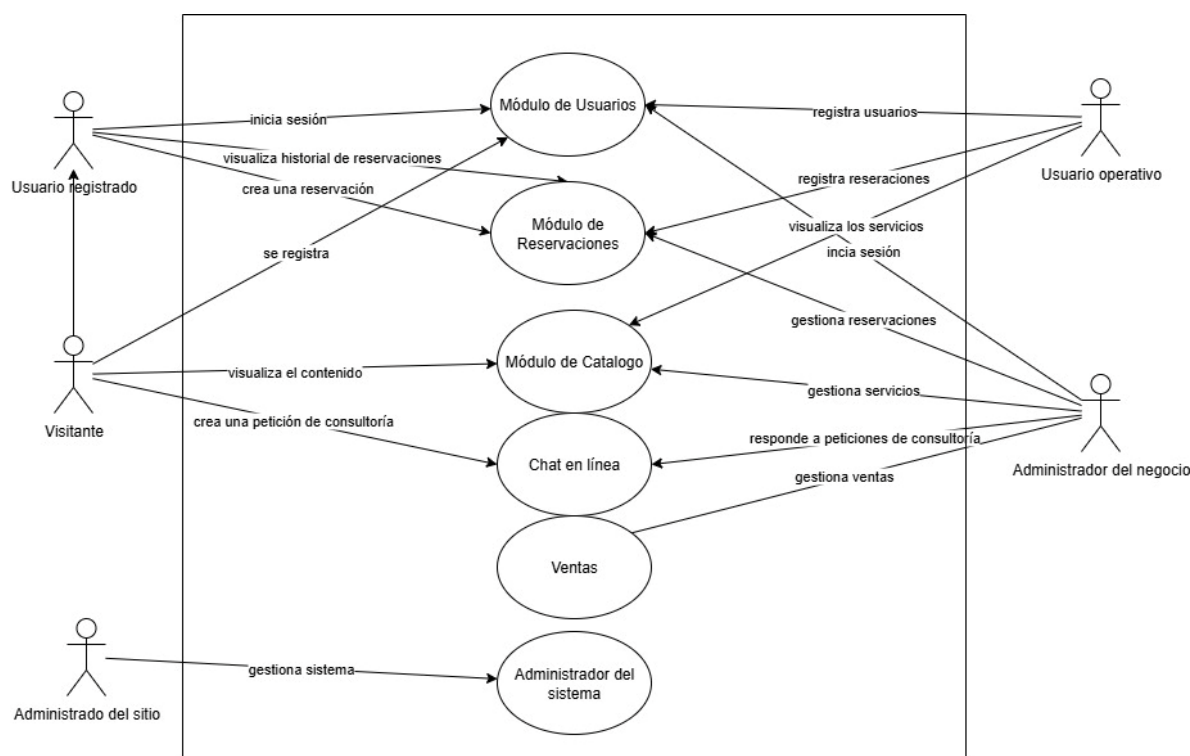


Figura 7: Diagrama general de casos de uso del Sistema Integral de Gestión Operativa Web

Descripción:

En este diagrama se observan los cinco actores principales (Usuario registrado, Usuario Operativo, Visitante, Administrador del sitio y Administrador del negocio) interactuando con el sistema central. Cada uno accede a diferentes grupos de casos de uso según su nivel de permisos:

el cliente realiza operaciones básicas como reservaciones y pagos; el administrador supervisa y gestiona los procesos internos; mientras que el desarrollador se encarga de la configuración y mantenimiento del sistema.

3.4.4. Diagramas de casos de uso específicos

Para detallar el comportamiento individual de los actores, se elaboraron tres diagramas complementarios que representan los casos de uso específicos de cada tipo de usuario.

3.4.4.1. *Casos de uso del administrador del negocio*

El administrador posee control total sobre la plataforma. Se encarga de la gestión de clientes, servicios, empleados y reportes, además de poder acceder a configuraciones institucionales.

Casos principales:

- Iniciar sesión
- Gestionar usuarios internos
- Gestionar servicios
- Consultar ventas
- Generar reportes en PDF
- Configurar parámetros del sistema
- Supervisar novedades
- Cerrar sesión

3.4.4.2. *Casos de uso del administrador del sitio*

El administrador del sitio tiene permisos avanzados que le permiten mantener y personalizar el sistema. Puede realizar exportaciones, respaldos, modificaciones de estructura o ajustes en la configuración global.

Casos principales:

- Iniciar sesión

- Configurar parámetros del sistema
- Exportar información
- Realizar respaldos
- Gestionar roles
- Supervisar novedades

3.4.4.3. Casos de uso del usuario operativo

El usuario operativo es un empleado de la empresa, este tiene menos accesos de configuración e información a comparación del administrador del negocio. A continuación, se enlistan los casos principales:

Casos principales:

- Iniciar sesión
- Registrar usuarios
- Registrar reservaciones
- Visualizar servicios
- Cambiar status de reservaciones

3.4.5. Descripción de casos de uso principales

En esta sección se presentan las descripciones detalladas de los principales casos de uso que conforman el sistema. Cada uno de ellos describe la interacción entre los actores y las funciones del sistema, especificando los objetivos, precondiciones, flujos de eventos y resultados esperados. Estas descripciones se elaboraron tomando como referencia el estándar UML 2.0 y constituyen la base para el diseño lógico y funcional de la aplicación.

Tabla 5 Caso de uso CU-01: Iniciar sesión

Elemento	Descripción
Actor principal	Todos los usuarios

Objetivo	Autenticar al usuario y permitir el acceso al sistema según su rol.
Entradas	Correo electrónico o usuario, contraseña.
Salidas	Acceso al panel correspondiente o mensaje de error.
Precondiciones	El usuario debe estar registrado.
Postcondiciones	Se crea una sesión activa con permisos definidos por el rol.
Flujo principal	1. El usuario ingresa sus credenciales. 2. El sistema verifica la información. 3. Se inicia sesión y se redirige al panel adecuado.
Flujo alternativo	Si las credenciales son incorrectas, el sistema muestra un mensaje de error.

Tabla 6 Caso de uso CU-02: Recuperar contraseña

Elemento	Descripción
Actor principal	Todos los usuarios
Objetivo	Permitir la recuperación del acceso mediante enlace de restablecimiento de contraseña.
Entradas	Correo electrónico registrado.
Salidas	Enlace de restablecimiento enviado al correo del usuario.
Precondiciones	El correo debe estar registrado en el sistema.
Postcondiciones	La nueva contraseña reemplaza la anterior y se actualiza el registro del usuario.
Flujo principal	1. El usuario solicita restablecer contraseña. 2. Ingresa su correo.

	3. El sistema envía un enlace temporal. 4. El usuario define una nueva contraseña.
Flujo alternativo	Si el correo no está registrado o el enlace expira, el sistema muestra un mensaje de error.

Tabla 7 Caso de uso CU-03: Gestionar clientes

Elemento	Descripción
Actor principal	Empleado / Administrador
Objetivo	Permitir registrar, modificar o eliminar la información de los clientes.
Entradas	Datos del cliente (nombre, correo, teléfono, etc.).
Salidas	Mensaje de confirmación o error según la operación.
Precondiciones	El usuario debe tener permisos internos.
Postcondiciones	La base de datos se actualiza con los cambios realizados.
Flujo principal	1. El usuario accede al módulo de clientes. 2. Registra, edita o elimina un cliente. 3. El sistema valida y guarda los cambios.
Flujo alternativo	Si la operación falla, el sistema muestra un mensaje de error sin afectar los registros.

Tabla 8 Caso de uso CU-04: Gestionar servicios

Elemento	Descripción
Actor principal	Administrador
Objetivo	Permitir crear, editar o eliminar los servicios ofrecidos por la empresa.
Entradas	Nombre, descripción, precio, imagen del servicio.

Salidas	Confirmación de operación realizada exitosamente.
Precondiciones	El usuario debe ser administrador.
Postcondiciones	El servicio se actualiza o se elimina de la base de datos.
Flujo principal	1. El administrador accede al módulo de servicios. 2. Realiza la operación correspondiente. 3. El sistema actualiza los registros.
Flujo alternativo	Si los datos son inválidos o la operación falla, el sistema muestra un mensaje de error.

Tabla 9 Caso de uso CU-5: Generar reportes

Elemento	Descripción
Actor principal	Administrador
Objetivo	Generar reportes administrativos en formato PDF.
Entradas	Selección del tipo de reporte (ventas, clientes, servicios).
Salidas	Archivo PDF generado y disponible para descarga.
Precondiciones	Debe existir información disponible en la base de datos.
Postcondiciones	El reporte se genera correctamente y se ofrece para descarga.
Flujo principal	1. El administrador elige el tipo de reporte. 2. El sistema recopila los datos. 3. Se genera el archivo PDF. 4. Se muestra el botón de descarga.
Flujo alternativo	Si no existen datos, el sistema muestra un mensaje indicando que no hay información disponible.

Tabla 10 Caso de uso CU-6: Configurar parámetros del sistema

Elemento	Descripción
Actor principal	Administrador / Desarrollador
Objetivo	Permitir modificar la información institucional del sistema.
Entradas	Nombre de empresa, logotipo, colores, enlaces, políticas.
Salidas	Mensaje de confirmación y cambios reflejados en la interfaz.
Precondiciones	El usuario debe tener permisos de administrador o desarrollador.
Postcondiciones	Los cambios se guardan en la base de datos y afectan la apariencia del sitio.
Flujo principal	1. El usuario accede a la configuración. 2. Modifica los valores. 3. El sistema valida y guarda los cambios. 4. Se actualiza la interfaz.
Flujo alternativo	Si se ingresan valores inválidos, el sistema cancela la operación y muestra un mensaje.

Tabla 11 Caso de uso CU-7: Exportar información

Elemento	Descripción
Actor principal	Desarrollador
Objetivo	Permitir exportar datos del sistema en formato CSV para respaldo o análisis.
Entradas	Selección del tipo de información a exportar.
Salidas	Archivo CSV generado y disponible para descarga.
Precondiciones	Debe existir información en la base de datos.

Postcondiciones	El sistema genera correctamente el archivo y lo pone a disposición del usuario.
Flujo principal	1. El desarrollador selecciona el tipo de exportación. 2. El sistema recopila los datos. 3. Se genera el archivo CSV. 4. Se habilita la descarga.
Flujo alternativo	Si no hay registros, el sistema muestra un mensaje informativo.

Tabla 12 Caso de uso CU-8: Registrar novedades

Elemento	Descripción
Actor principal	Sistema
Objetivo	Registrar automáticamente las acciones relevantes realizadas por los usuarios internos.
Entradas	Identificador del usuario, acción realizada, fecha y hora.
Salidas	Registro almacenado en la bitácora del sistema.
Precondiciones	Debe haberse ejecutado una acción relevante.
Postcondiciones	La acción queda registrada para auditoría.
Flujo principal	1. El usuario interno realiza una acción. 2. El sistema captura la información. 3. Se almacena en la base de datos.
Flujo alternativo	Si la conexión falla, el registro se almacena en cola hasta que se restablezca la comunicación

Tabla 13 Caso de uso CU-9: Cerrar sesión

Elemento	Descripción
-----------------	--------------------

Actor principal	Todos los usuarios
Objetivo	Finalizar de manera segura la sesión activa del usuario.
Entradas	Acción de cierre de sesión.
Salidas	Mensaje de confirmación y redirección al inicio.
Precondiciones	El usuario debe tener una sesión activa.
Postcondiciones	La sesión se destruye y el usuario regresa a la página principal.
Flujo principal	1. El usuario selecciona “Cerrar sesión”. 2. El sistema elimina la sesión activa. 3. Redirige al inicio.
Flujo alternativo	Si la sesión ya expiró, el sistema redirige directamente a la pantalla de login.

3.5. Modelado del Sistema

El modelado del sistema tiene como propósito representar la estructura lógica, las relaciones internas y el comportamiento dinámico del software antes de su implementación. Esta etapa permite transformar los requerimientos funcionales y los casos de uso previamente definidos en modelos técnicos que describen cómo está organizado el sistema y cómo interactúan sus diferentes componentes.

El modelado se realizó utilizando el lenguaje UML (Unified Modeling Language), estándar en la ingeniería de software para representar sistemas orientados a objetos. Se emplearon herramientas como StarUML y Lucidchart para elaborar los diagramas, asegurando claridad y consistencia en las representaciones visuales.

A continuación, se presentan los principales modelos que conforman la estructura del sistema: el **diagrama de clases**, el **modelo entidad-relación**, los **diagramas de secuencia**, el **diagrama de componentes** y el **diagrama de despliegue**, los cuales proporcionan una visión integral tanto lógica como física de la aplicación.

3.5.1. Diagrama de clases

El diagrama de clases es el modelo central en el diseño orientado a objetos, ya que define las entidades principales, sus atributos, métodos y relaciones. En este sistema, las clases representan los elementos que conforman la lógica de negocio, tales como usuarios, servicios, reservaciones, pagos y reportes.

Cada clase fue diseñada considerando los principios de encapsulamiento, modularidad y reutilización. Las relaciones entre ellas fueron definidas mediante asociaciones, composiciones y herencias, según el tipo de vínculo existente entre los objetos.

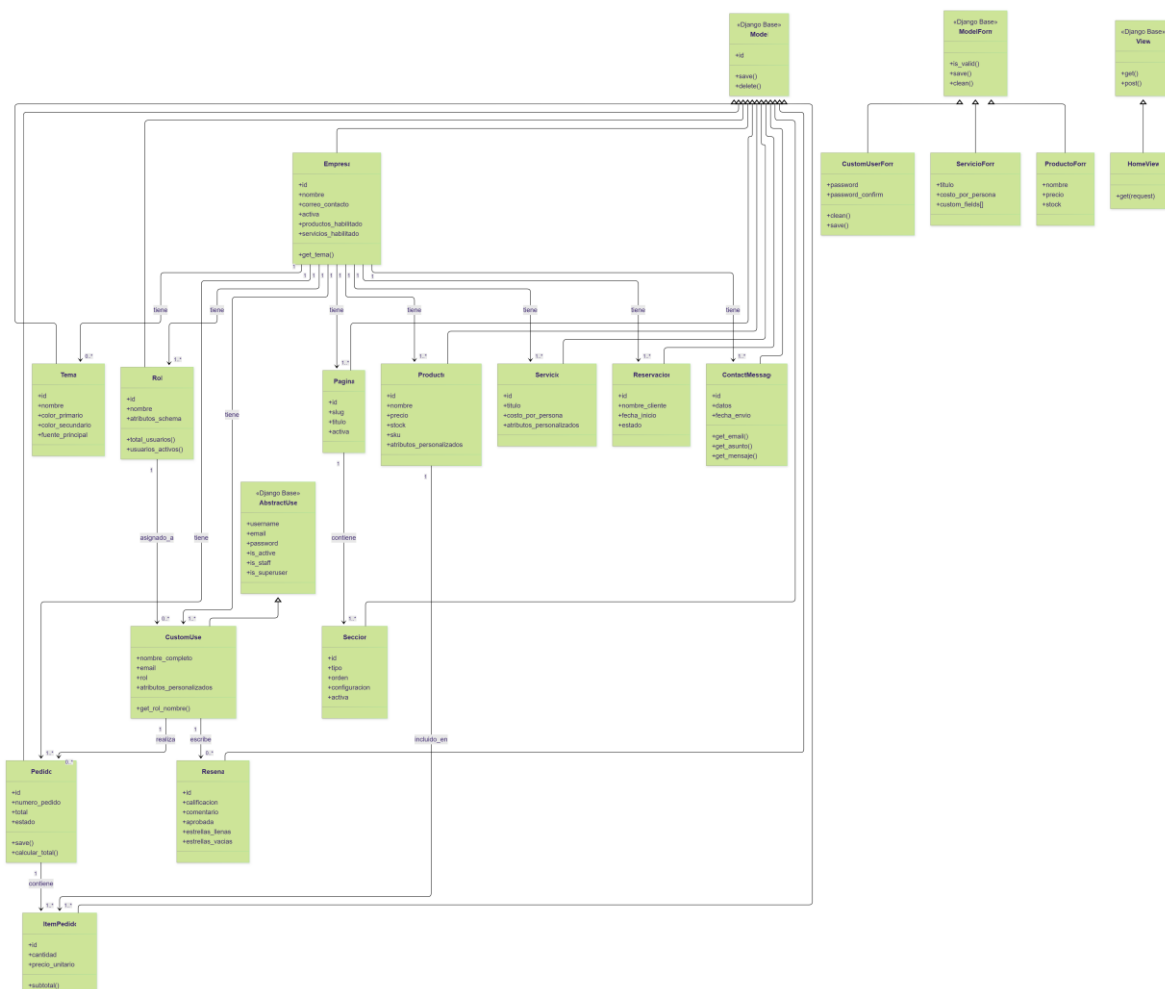


Figura 8 Diagrama de clases

Descripción:

El diagrama de clases está compuesto por los siguientes elementos principales:

- **Clase Usuario:** representa a las personas que interactúan con el sistema. Contiene atributos como nombre, correo, contraseña, rol y estado. Incluye métodos para autenticación, actualización de datos y control de acceso.
- **Clase Rol:** define los permisos y privilegios asociados a cada usuario (cliente, empleado, administrador, desarrollador).
- **Clase Servicio:** almacena la información de los servicios disponibles (nombre, descripción, costo, imagen y estado).
- **Clase Reservación:** relaciona al cliente con un servicio en una fecha y hora determinadas. Posee atributos como id_reservacion, fecha, hora, estado y total.
- **Clase Pago:** gestiona la información de los pagos efectuados por los clientes, incluyendo monto, fecha_pago, método y estatus.
- **Clase Reporte:** permite generar documentos administrativos con base en los registros de ventas, usuarios o servicios.
- **Clase Configuración:** almacena la información institucional y visual del sistema (nombre, logotipo, colores, enlaces).
- **Clase Novedad:** registra las acciones realizadas por los usuarios internos para fines de auditoría.

Estas clases interactúan mediante relaciones **uno a muchos (1:N)**, como Usuario–Reservación, y **uno a uno (1:1)** entre Reservación–Pago. Además, la clase “Rol” tiene una relación de **dependencia con Usuario**, lo que permite mantener un control jerárquico y seguro de los permisos del sistema.

El diagrama de clases establece la base estructural sobre la cual se implementara el código en Django, donde cada clase se traduce en un modelo dentro del ORM, facilitando la persistencia y manipulación de datos.

3.5.2. Modelos

Base: django.db.models.Model

- Todas las clases de modelo heredan de esta clase base

Empresa (models.Model)

Atributos:

- nombre, nombre_titular, correo_contacto, telefono, ubicacion
- cuenta_bancaria, clabe, numero_terjeta
- logotipo, sitio_web, subdominio, dominio_personalizado
- activa, productos_habilitado, servicios_habilitado, resenas_habilitado
- smtp_host, smtp_port, smtp_user, smtp_password, smtp_use_tls, smtp_use_ssl
- fecha_creacion, creado_en

Métodos:

- get_tema() - Retorna el tema de la empresa o None

Relaciones:

- ForeignKey a Tema (opcional)
- Relaciones inversas: usuarios, roles, paginas, productos, servicios, etc.

Tema (models.Model)

Atributos:

- nombre (unique)
- color_primario, color_secundario, color_acento, color_texto, color_fondo
- fuente_principal, fuente_secundaria
- estilo_layout, navbar_estilo
- logo_header, imagen_hero, favicon
- activo, creado_en, actualizado_en

CustomUser (AbstractUser)

Herencia: django.contrib.auth.models.AbstractUser

Atributos adicionales:

- nombre_completo
- email (unique)
- fecha_nacimiento, telefono
- empresa (ForeignKey)
- rol_legacy (deprecated)
- rol (ForeignKey a Rol)
- atributos_personalizados (JSON)

Métodos:

- get_rol_nombre() - Retorna el nombre del rol actual

Relaciones:

- ForeignKey a Empresa
- ForeignKey a Rol
- Relaciones inversas: pedidos, resenas, novedades

Rol (models.Model)

Atributos:

- nombre, descripcion
- empresa (ForeignKey)
- atributos_schema (JSON)
- permisos (JSON)
- activo, creado_en, actualizado_en

Métodos:

- total_usuarios() - Cuenta usuarios con este rol
- usuarios_activos() - Cuenta usuarios activos con este rol

Relaciones:

- ForeignKey a Empresa
- Relación inversa: usuarios

Pagina (models.Model)

Atributos:

- empresa (ForeignKey)
- slug (choices: home, servicios, productos, nosotros, galeria, contacto, footer)
- titulo, activa
- creado_en, actualizado_en

Restricciones:

- unique_together: (empresa, slug)

Relaciones:

- ForeignKey a Empresa
- Relación inversa: secciones

Seccion (models.Model)

Atributos:

- pagina (ForeignKey)
- tipo (choices: hero, texto, carousel, galeria, features, cta, testimonios, mapa, instagram, contacto_form, servicios_accordion)
- titulo, orden, activa
- configuracion (JSON)
- creado_en, actualizado_en

Relaciones:

- ForeignKey a Pagina

Producto (models.Model)

Atributos:

- empresa (ForeignKey)
- nombre, descripcion
- precio, stock, sku (unique)
- categoria (ForeignKey, nullable)

- perecedero, fecha_caducidad
- imagen, activo
- atributos_personalizados (JSON)
- creado_en

Relaciones:

- ForeignKey a Empresa
- ForeignKey a CategoriaProducto
- Relación inversa: items_pedido, resenas (genérico)

Pedido (models.Model)

Atributos:

- empresa (ForeignKey)
- usuario (ForeignKey, nullable)
- nombre_cliente, email_cliente, telefono_cliente
- direccion_entrega, notas
- numero_pedido (unique, auto-generado)
- total, estado, metodo_pago
- fecha_pedido, fecha_actualizacion, fecha_entrega_estimada

Métodos:

- save() - Genera número de pedido único automáticamente
- calcular_total() - Calcula total sumando items

Relaciones:

- ForeignKey a Empresa
- ForeignKey a CustomUser
- Relación inversa: items

Resena (models.Model)

Atributos:

- usuario (ForeignKey)

- content_type (ForeignKey a ContentType)
- object_id (PositiveIntegerField)
- calificacion (1-5)
- comentario
- aprobada
- fecha_creacion, fecha_actualizacion

Relaciones:

- ForeignKey a CustomUser
- GenericForeignKey (Producto o Servicio)

Métodos:

- estrellas_llenas (property)
- estrellas_vacias (property)

Servicio (models.Model)

Atributos:

- empresa (ForeignKey)
- servicio (ForeignKey a TipoServicio)
- titulo, descripcion
- costo_por_persona, costo_niño, costo_con_descuento
- imagen_principal, duracion, restricciones
- atributos_personalizados (JSON)

Relaciones:

- ForeignKey a Empresa
- ForeignKey a TipoServicio
- Relación inversa: imagenes, reservacion_servicio, resenas (genérico)

Reservacion (models.Model)

Atributos:

- empresa (ForeignKey)

- nombre_cliente, email_cliente
- fecha_inicio, fecha_fin, hora_recepcion
- numero_adultos, numero_ninos, numero_descuento
- fecha_reserva, comentario
- pago_realizado, total_pagado
- comprobante_pago, estado

Relaciones:

- ForeignKey a Empresa
- Relación inversa: reservacion_servicio

ContactMessage (models.Model)

Atributos:

- empresa (ForeignKey)
- datos (JSON)
- fecha_envio, leído, respondido
- notas_internas
- ip_address, user_agent

Métodos:

- get_email()
- get_asunto()
- get_mensaje()

Relaciones:

- ForeignKey a Empresa

3.5.3. Formularios (Forms)

Base: django.forms.ModelForm

- Todos los formularios heredan de ModelForm

CustomUserForm (ModelForm)

Modelo: CustomUser

Campos:

- username, email, rol, is_active, telefono
- password, password_confirm

Métodos:

- __init__(empresa) - Filtra roles por empresa
- clean() - Valida que passwords coincidan
- save() - Guarda usuario con password hashado

ServicioForm (ModelForm)

Modelo: Servicio

Campos:

- titulo, servicio, descripcion
- costo_por_persona, costo_niño, costo_con_descuento
- imagen_principal, duracion, restricciones
- Campos personalizados dinámicos

Métodos:

- __init__(empresa) - Agrega campos personalizados según atributos de la empresa

ProductoForm (ModelForm)

Modelo: Producto

Campos:

- nombre, descripcion, precio, stock
- categoria, percedero, fecha_caducidad
- imagen, atributos_personalizados
- Campos personalizados dinámicos

EmpresaForm (ModelForm)

Modelo: Empresa

Campos:

- Información básica, branding, SMTP, módulos

*SeccionConfigForm (Form)***Campos:**

- configuracion_json (JSONField)

Métodos:

- clean() - Valida JSON

3.5.4. Vistas (Views)

HomeView (View)

Clase base: `django.views.View`

Métodos:

- get(request) - Renderiza página de inicio

*Funciones de Vista (Function-Based Views)***home/views.py:**

- index(request) - Página principal con secciones dinámicas
- nosotros(request) - Página nosotros
- productos(request) - Catálogo de productos
- producto_detalle(request, producto_id) - Detalle de producto
- agregar_al_carrito(request, producto_id) - Agregar a carrito
- ver_carrito(request) - Vista del carrito
- checkout(request) - Proceso de pago
- procesar_pedido(request) - Procesa pedido y envía email
- procesar_contacto(request) - Procesa formulario de contacto
- get_pagina_secciones(slug_pagina) - Helper function

adminpanel/views.py:

- dashboard(request) - Dashboard principal
- lista_usuarios(request) - Lista de usuarios
- agregar_usuario(request) - Crear usuario
- editar_usuario(request, id) - Editar usuario
- admin_productos(request) - Lista de productos
- admin_agregar_producto(request) - Crear producto
- editar_producto(request, pk) - Editar producto
- exportar_productos_pdf(request) - Exportar a PDF

devpanel/views.py:

- gestion_empresas(request) - Gestión de empresas
- crear_empresa(request) - Crear empresa
- editar_empresa(request, pk) - Editar empresa
- gestion_secciones(request) - Gestión de secciones
- crear_seccion(request, slug_pagina) - Crear sección
- editar_seccion(request, pk) - Editar sección
- gestion_rolles(request) - Gestión de roles

usuarios/views.py:

- registro_view(request) - Registro de usuarios
- login_view(request) - Login
- logout_view(request) - Logout

3.5.5. Utilidades y Helpers

Context Processors

home/context_processors.py:

- carrito_context(request) - Añade contador de carrito
- footer_context(request) - Añade secciones del footer

adminpanel/context_processors.py:

- empresa_context(request) - Añade empresa activa y tema

devpanel/context_processors.py:

- `empresa_activa_processor(request)` - Añade nombre de empresa activa

Management Commands

devpanel/management/commands/:

- `seed_xoletongo_content.py` - Seeder para contenido inicial
- `seed_nosotros_content.py` - Seeder para página nosotros
- `crear_pagina_footer.py` - Crea página de footer
- `crear_pagina_productos.py` - Crea página de productos

Base: `django.core.management.base.BaseCommand`

Métodos:

- `handle(*args, **options)` - Ejecuta el comando

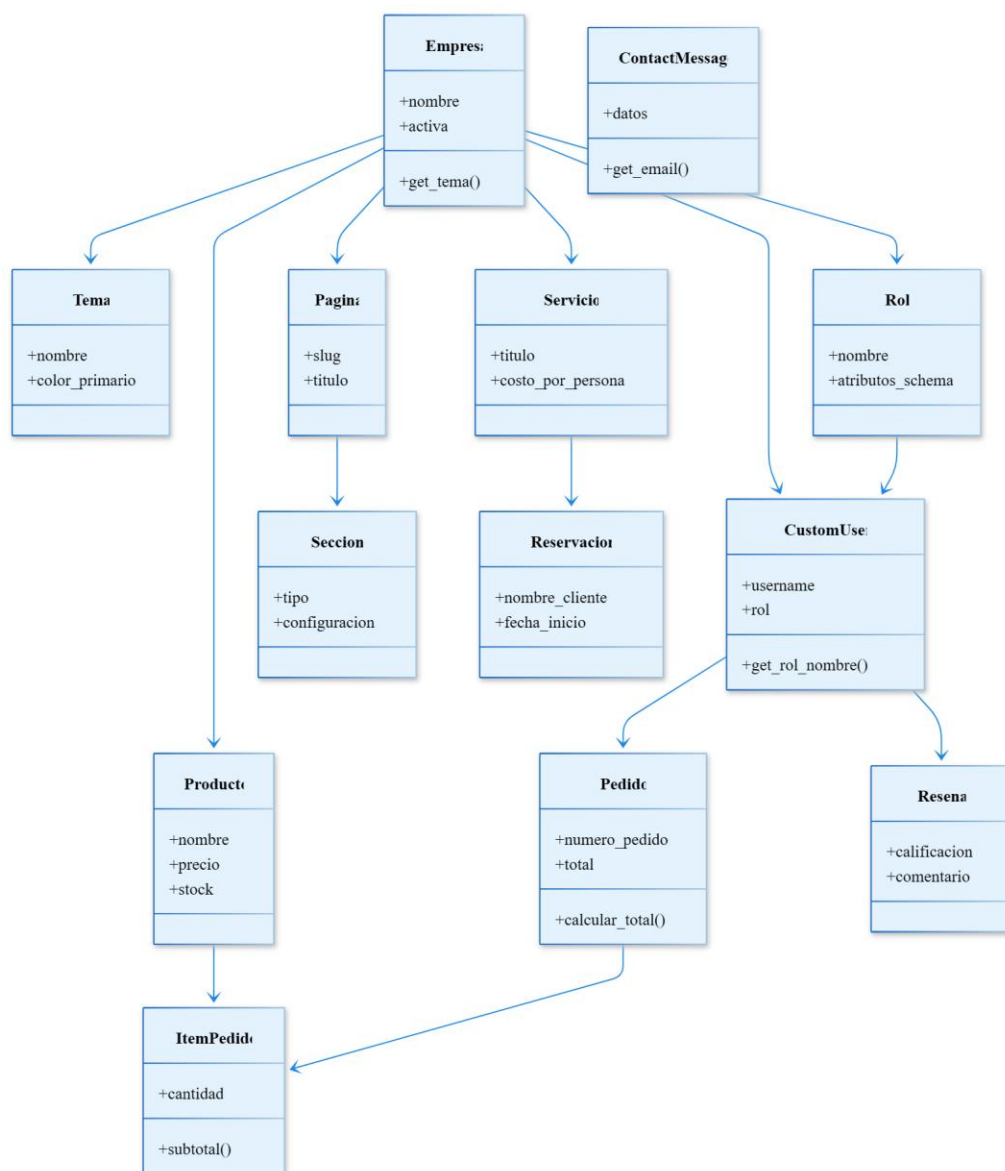


Figura 9 Diagrama de clases simplificado por módulos

Tabla 14 Comparación: Diagrama ER vs Diagrama de Clases

Aspecto	Diagrama ER	Diagrama de Clases
Propósito	Estructura de BD	Estructura de código
Enfoque	Tablas y relaciones	Clases y métodos
Incluye	Atributos de BD	Métodos, propiedades
Relaciones	Foreign Keys	Herencia, composición
Campos	Columnas de BD	Atributos y métodos

3.5.6 Diagrama entidad–relación (E–R)

El diagrama entidad–relación representa el modelo lógico de la base de datos del sistema, mostrando las tablas, atributos, claves primarias y relaciones entre ellas. Este modelo asegura la integridad referencial de los datos y facilita la comprensión del almacenamiento de información.

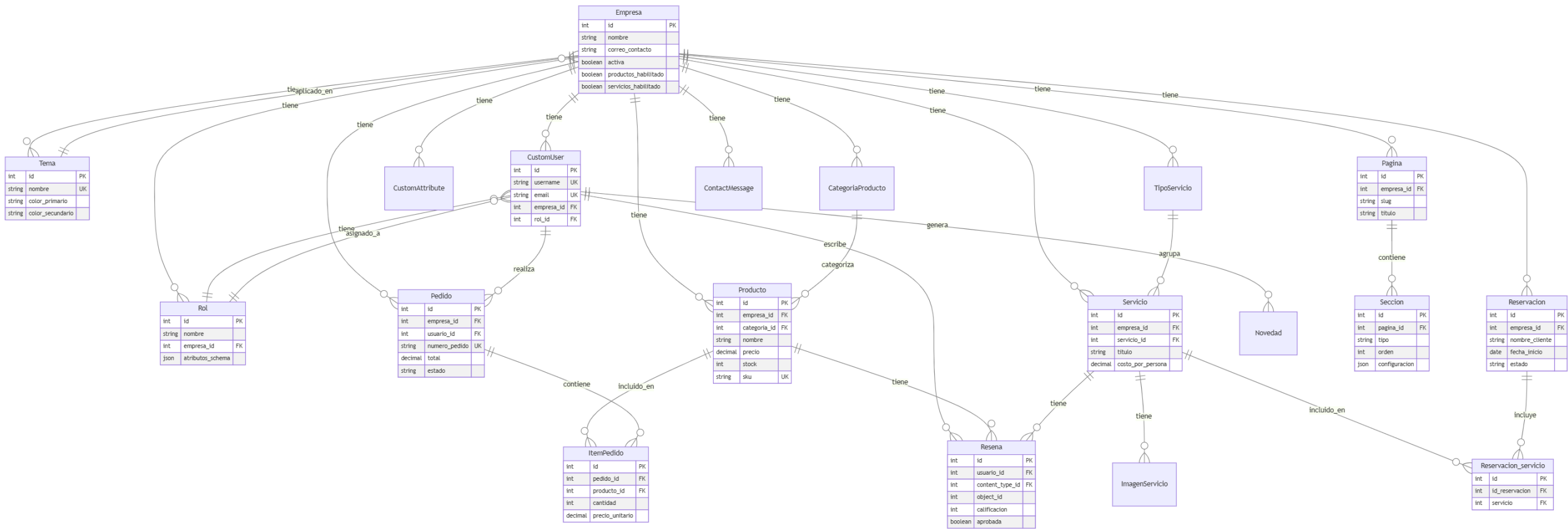


Figura 10 Diagrama Entidad-Relación

Descripción:

El modelo E–R está compuesto por las siguientes entidades:

- **Usuarios:** contiene los datos personales y credenciales de acceso. Su clave primaria es `id_usuario`.
- **Roles:** define los tipos de usuario y sus permisos (`id_rol`, `nombre_rol`).
- **Servicios:** almacena la información sobre los servicios ofrecidos (`id_servicio`, `nombre`, `descripcion`, `precio`, `imagen`).
- **Reservaciones:** registra las solicitudes generadas por los clientes (`id_reservacion`, `fecha`, `hora`, `estado`, `id_usuario`, `id_servicio`).
- **Pagos:** guarda la información de los pagos asociados a reservaciones (`id_pago`, `monto`, `metodo`, `id_reservacion`).
- **Reportes:** almacena la información generada para fines administrativos (`id_reporte`, `tipo`, `fecha_generacion`, `id_usuario`).
- **Novedades:** contiene el registro de acciones ejecutadas por usuarios internos (`id_novedad`, `descripcion`, `fecha`, `id_usuario`).
- **Configuración:** guarda los datos de personalización del sistema (`id_config`, `nombre_empresa`, `logo`, `color_primario`, `color_secundario`, `politicas`).

Relaciones principales:

- Un **rol** puede estar asociado a muchos **usuarios** (1:N).
- Un **usuario** puede generar múltiples **reservaciones** (1:N).
- Cada **reservación** está vinculada a un **servicio** (N:1) y puede tener un único **pago** (1:1).
- Un **administrador** o **desarrollador** puede generar muchos **reportes** (1:N).
- Cada **novedad** está asociada a un usuario responsable (N:1).

Este modelo garantiza la consistencia y normalización de los datos hasta la **tercera forma normal (3FN)**, evitando redundancias y asegurando un desempeño óptimo de la base de datos.

3.5.6.1 Entidades Principales

Empresa (Entidad central - Multi-tenant)

Atributos:

- id (PK)
- nombre
- nombre_titular
- correo_contacto
- telefono
- ubicacion
- cuenta_bancaria
- clabe
- numero_tarjeta
- logotipo (imagen)
- sitio_web
- subdominio (unique)
- dominio_personalizado (unique)
- activa (boolean)
- fecha_creacion
- creado_en
- productos_habilitado (boolean)
- servicios_habilitado (boolean)
- resenas_habilitado (boolean)
- smtp_host
- smtp_port

- smtp_user
- smtp_password
- smtp_use_tls (boolean)
- smtp_use_ssl (boolean)

Tema

Atributos:

- id (PK)
- nombre (unique)
- color_primario
- color_secundario
- color_acento
- color_texto
- color_fondo
- fuente_principal
- fuente_secundaria
- estilo_layout
- navbar_estilo
- logo_header (imagen)
- imagen_hero (imagen)
- favicon (imagen)
- activo (boolean)
- creado_en
- actualizado_en

CustomUser (Usuario personalizado)

Atributos:

- id (PK) - hereda de AbstractUser
- username (unique)
- email (unique)
- password
- nombre_completo
- fecha_nacimiento
- telefono
- rol_legacy (deprecated)
- atributos_personalizados (JSON)
- is_active
- is_staff
- is_superuser
- date_joined
- last_login

Rol (Roles dinámicos)

Atributos:

- id (PK)
- nombre
- descripcion
- atributos_schema (JSON)
- permisos (JSON)
- activo (boolean)
- creado_en
- actualizado_en

Pagina

Atributos:

- id (PK)
- slug (home, servicios, productos, nosotros, galeria, contacto, footer)
- titulo
- activa (boolean)
- creado_en
- actualizado_en

Seccion

Atributos:

- id (PK)
- tipo (hero, texto, carousel, galeria, features, cta, testimonios, mapa, instagram, contacto_form, servicios_accordion)
- titulo
- orden
- activa (boolean)
- configuracion (JSON)
- creado_en
- actualizado_en

CustomAttribute

Atributos:

- id (PK)
- target_model (Producto, Servicio)
- name

- attribute_type (TEXT, TEXTAREA, NUMBER, DATE, BOOLEAN)

CategoriaProducto

Atributos:

- id (PK)
- nombre
- descripcion

Producto

Atributos:

- id (PK)
- nombre
- descripcion
- precio
- perecedero (boolean)
- fecha_caducidad
- imagen
- stock
- sku (unique)
- activo (boolean)
- atributos_personalizados (JSON)
- creado_en

Pedido

Atributos:

- id (PK)
- numero_pedido (unique)

- nombre_cliente
- email_cliente
- telefono_cliente
- direccion_entrega
- notas
- total
- estado (pendiente, confirmado, en_proceso, entregado, cancelado)
- metodo_pago (efectivo, transferencia, tarjeta)
- fecha_pedido
- fecha_actualizacion
- fecha_entrega_estimada

ItemPedido

Atributos:

- id (PK)
- nombre_producto
- precio_unitario
- cantidad

TipoServicio

Atributos:

- id (PK)
- nombre
- descripcion
- tipo (porDia, porHora)

Servicio

Atributos:

- id (PK)
- titulo
- descripcion
- costo_por_persona
- costo_niño
- costo_con_descuento
- imagen_principal
- duracion
- restricciones
- atributos_personalizados (JSON)

ImagenServicio

Atributos:

- id (PK)
- imagen
- descripcion
- orden

Reservacion

Atributos:

- id (PK)
- nombre_cliente
- email_cliente
- fecha_inicio

- fecha_fin
- hora_recepcion
- numero_adultos
- numero_ninos
- numero_descuento
- fecha_reserva
- comentario
- pago_realizado (boolean)
- total_pagado
- comprobante_pago (archivo)
- estado (pendiente, aprobada, finalizada)

Reservacion_servicio (Tabla intermedia)

Atributos:

- id (PK)

Resena (Genérico - para Productos y Servicios)

Atributos:

- id (PK)
- calificacion (1-5)
- comentario
- fecha_creacion
- fecha_actualizacion
- aprobada (boolean)
- content_type_id (FK a ContentType)
- object_id (ID del objeto relacionado)

ContactMessage

Atributos:

- id (PK)
- datos (JSON)
- fecha_envio
- leído (boolean)
- respondido (boolean)
- notas_internas
- ip_address
- user_agent

Venta

Atributos:

- id (PK)
- titulo
- monto
- fecha
- estado (pendiente, pagado, cancelado)

Novedad (Log de actividades)

Atributos:

- id (PK)
- accion
- fecha

3.5.6.2. Relaciones y Cardinalidades

Relaciones Directas

1. **Empresa** → **Tema** (1:N)
 - Una empresa tiene un tema (opcional)
 - Un tema puede pertenecer a múltiples empresas
 - Relación: Empresa.tema → Tema.id (FK, nullable)
2. **Empresa** → **CustomUser** (1:N)
 - Una empresa tiene muchos usuarios
 - Un usuario pertenece a una empresa (opcional)
 - Relación: CustomUser.empresa → Empresa.id (FK, nullable)
3. **Empresa** → **Rol** (1:N)
 - Una empresa tiene muchos roles
 - Un rol pertenece a una empresa
 - Relación: Rol.empresa → Empresa.id (FK)
4. **CustomUser** → **Rol** (N:1)
 - Muchos usuarios tienen un rol
 - Un rol tiene muchos usuarios
 - Relación: CustomUser.rol → Rol.id (FK, nullable)
5. **Empresa** → **Pagina** (1:N)
 - Una empresa tiene muchas páginas
 - Una página pertenece a una empresa
 - Relación: Pagina.empresa → Empresa.id (FK)
 - **Restricción:** unique_together (empresa, slug)
6. **Pagina** → **Seccion** (1:N)
 - Una página tiene muchas secciones
 - Una sección pertenece a una página
 - Relación: Seccion.pagina → Pagina.id (FK)
7. **Empresa** → **CustomAttribute** (1:N)
 - Una empresa tiene muchos atributos personalizados
 - Un atributo pertenece a una empresa

- Relación: CustomAttribute.empresa → Empresa.id (FK)
 - **Restricción:** unique_together (empresa, target_model, name)
8. **Empresa → CategoriaProducto (1:N)**
- Una empresa tiene muchas categorías
 - Una categoría pertenece a una empresa
 - Relación: CategoriaProducto.empresa → Empresa.id (FK)
9. **CategoriaProducto → Producto (1:N)**
- Una categoría tiene muchos productos
 - Un producto pertenece a una categoría (opcional)
 - Relación: Producto.categoria → CategoriaProducto.id (FK, nullable)
10. **Empresa → Producto (1:N)**
- Una empresa tiene muchos productos
 - Un producto pertenece a una empresa
 - Relación: Producto.empresa → Empresa.id (FK)
11. **Empresa → Pedido (1:N)**
- Una empresa tiene muchos pedidos
 - Un pedido pertenece a una empresa
 - Relación: Pedido.empresa → Empresa.id (FK)
12. **CustomUser → Pedido (1:N)**
- Un usuario puede tener muchos pedidos
 - Un pedido puede tener un usuario (opcional - también permite guest)
 - Relación: Pedido.usuario → CustomUser.id (FK, nullable)
13. **Pedido → ItemPedido (1:N)**
- Un pedido tiene muchos items
 - Un item pertenece a un pedido
 - Relación: ItemPedido.pedido → Pedido.id (FK)
14. **Producto → ItemPedido (1:N)**
- Un producto puede estar en muchos items de pedido
 - Un item de pedido referencia un producto
 - Relación: ItemPedido.producto → Producto.id (FK, PROTECT)

15. Empresa → TipoServicio (1:N)

- Una empresa tiene muchos tipos de servicio
- Un tipo de servicio pertenece a una empresa
- Relación: TipoServicio.empresa → Empresa.id (FK)

16. TipoServicio → Servicio (1:N)

- Un tipo de servicio tiene muchos servicios
- Un servicio pertenece a un tipo de servicio
- Relación: Servicio.servicio → TipoServicio.id (FK)

17. Empresa → Servicio (1:N)

- Una empresa tiene muchos servicios
- Un servicio pertenece a una empresa
- Relación: Servicio.empresa → Empresa.id (FK)

18. Servicio → ImagenServicio (1:N)

- Un servicio tiene muchas imágenes
- Una imagen pertenece a un servicio
- Relación: ImagenServicio.servicio → Servicio.id (FK)

19. Empresa → Reservacion (1:N)

- Una empresa tiene muchas reservaciones
- Una reservación pertenece a una empresa
- Relación: Reservacion.empresa → Empresa.id (FK)

20. Reservacion → Reservacion_servicio (1:N)

- Una reservación tiene muchos servicios asociados (tabla intermedia)
- Una reservación-servicio pertenece a una reservación
- Relación: Reservacion_servicio.id_reservacion → Reservacion.id (FK)

21. Servicio → Reservacion_servicio (1:N)

- Un servicio puede estar en muchas reservaciones
- Una reservación-servicio referencia un servicio
- Relación: Reservacion_servicio.servicio → Servicio.id (FK)

22. CustomUser → Resena (1:N)

- Un usuario puede hacer muchas reseñas
- Una reseña pertenece a un usuario
- Relación: Resena.usuario → CustomUser.id (FK)

23. **Resena** → **Producto/Servicio** (N:1) - **Relación Genérica**

- Muchas reseñas pueden ser para un producto O servicio
- Una reseña es para un producto O servicio
- Relación: Resena.content_type + Resena.object_id → Producto.id O Servicio.id
- **Restricción:** unique_together (usuario, content_type, object_id)

24. **Empresa** → **ContactMessage** (1:N)

- Una empresa tiene muchos mensajes de contacto
- Un mensaje pertenece a una empresa
- Relación: ContactMessage.empresa → Empresa.id (FK)

25. **CustomUser** → **Novedad** (1:N)

- Un usuario genera muchas novedades (log de actividades)
- Una novedad pertenece a un usuario
- Relación: Novedad.usuario → CustomUser.id (FK)

3.5.6.3. *Especificaciones del sistema*

Entidad Central: Empresa

- **Empresa** es la entidad central del sistema multi-tenant
- Casi todas las entidades principales tienen relación con Empresa
- Permite aislar datos entre diferentes empresas

Relaciones Genéricas

- **Resena** usa ContentType para relacionarse con Producto O Servicio
- Esto permite que un mismo modelo de reseña funcione para ambos tipos

Tablas Intermedias

- **Reservacion_servicio** es una tabla intermedia (many-to-many)
- Conecta Reservacion con Servicio (una reserva puede tener múltiples servicios)

Campos JSON

- Varias entidades usan campos JSON para flexibilidad:
 - Seccion.configuracion - Configuración dinámica de secciones
 - Producto.atributos_personalizados - Atributos custom
 - Servicio.atributos_personalizados - Atributos custom
 - CustomUser.atributos_personalizados - Valores de atributos del rol
 - Rol.atributos_schema - Definición de atributos del rol
 - Rol.permisos - Permisos del rol (futuro)
 - ContactMessage.datos - Datos dinámicos del formulario

Soft Delete / Estado Activo

- Muchas entidades tienen campo activo o activa para soft delete
- Permite desactivar sin eliminar registros

Campos de Auditoría

- creado_en, actualizado_en, fecha_creacion en múltiples entidades
- Permite tracking de cambios

3.5.7. Diagrama de secuencia

El diagrama de secuencia describe la interacción entre los objetos del sistema a lo largo del tiempo, mostrando el flujo de mensajes que se intercambian para ejecutar un proceso específico.

3.5.7.1. Diagrama de secuencia – Empleado

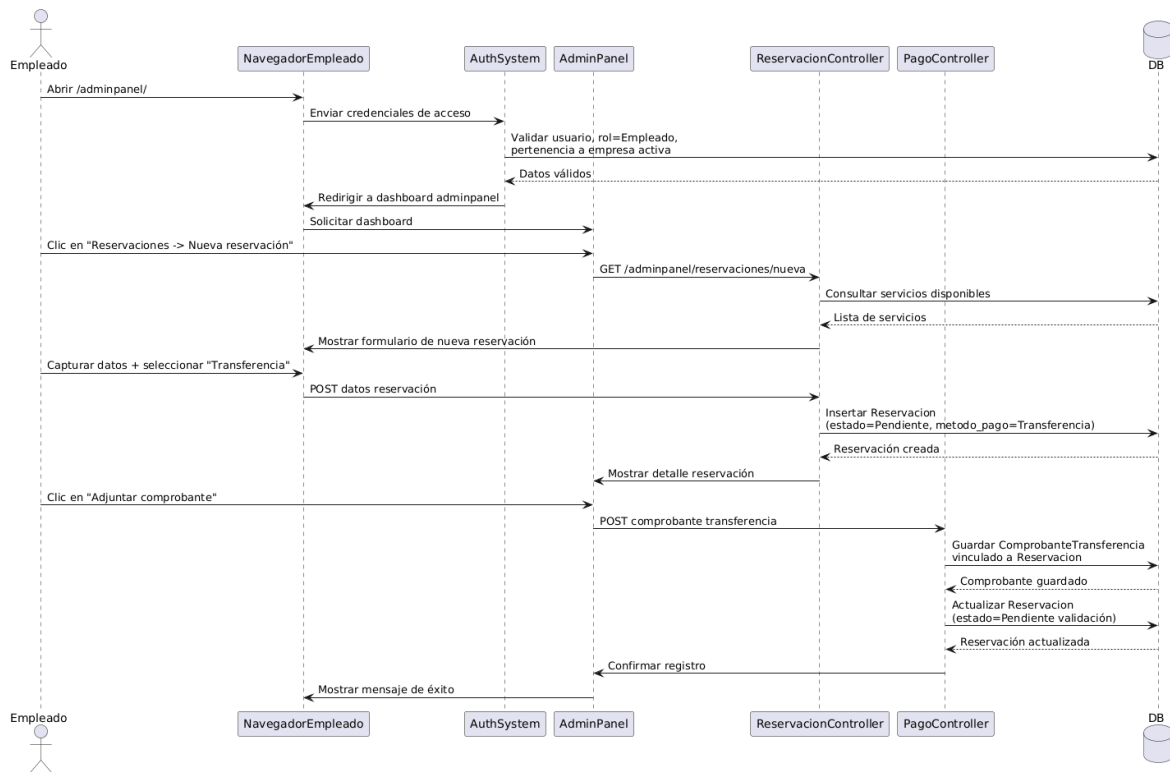


Figura 11 Diagrama de secuencia - Empleado

Escenario: registrar una reservación interna y subir comprobante de transferencia

Basado en tu documento:

- El **Empleado** puede registrar reservaciones y subir comprobantes, pero no editar la información de la empresa.

proyecto final

- Lo hace desde el **panel administrativo interno (adminpanel)**.

Participantes (lifelines)

- Empleado (actor)
- NavegadorEmpleado (opcional, solo para distinguir cliente web)
- AdminPanel (vista/controlador Django en adminpanel/)
- AuthSystem (módulo de autenticación Django)
- ReservacionController (vista que gestiona reservaciones internas)
- PagoController (vista/servicio para registrar comprobantes)

- DB (base de datos: tablas de Reservacion y ComprobanteTransferencia)

Flujo narrado

1. El **Empleado** abre el navegador y entra a /adminpanel/.
2. El **NavegadorEmpleado** envía petición de login al **AuthSystem**.
3. El **AuthSystem** valida credenciales y rol (Empleado) y verifica que pertenece a la empresa activa.
4. Si es correcto, el **AuthSystem** redirige al **AdminPanel** (dashboard interno).
5. El Empleado selecciona la opción “**Reservaciones** → **Nueva reservación**”.
6. El **AdminPanel** envía la petición a ReservacionController.
7. ReservacionController consulta a la **DB** los servicios disponibles de la empresa activa y devuelve el formulario de creación de reservación.
8. El Empleado llena datos del cliente/servicio/fechas y selecciona **método de pago: Transferencia bancaria**.
9. El formulario se envía a ReservacionController, que valida la información.
10. ReservacionController crea un registro en **DB** (tabla Reservacion) con estado Pendiente y método_pago = Transferencia.
11. El **AdminPanel** muestra la pantalla de detalle de la reservación, con la opción de “**Adjuntar comprobante**”.
12. El Empleado selecciona un archivo (imagen/PDF) y envía el formulario.
13. El PagoController recibe la petición, valida el archivo y registra el **comprobante de transferencia** en la **DB**, vinculado a la reservación.
14. PagoController actualiza la reservación a un estado intermedio (por ejemplo, Pendiente validación).
15. El **AdminPanel** responde mostrando mensaje de éxito y el Empleado visualiza la reservación con su comprobante asociado.

3.5.7.2. Diagrama de secuencia – Administrador

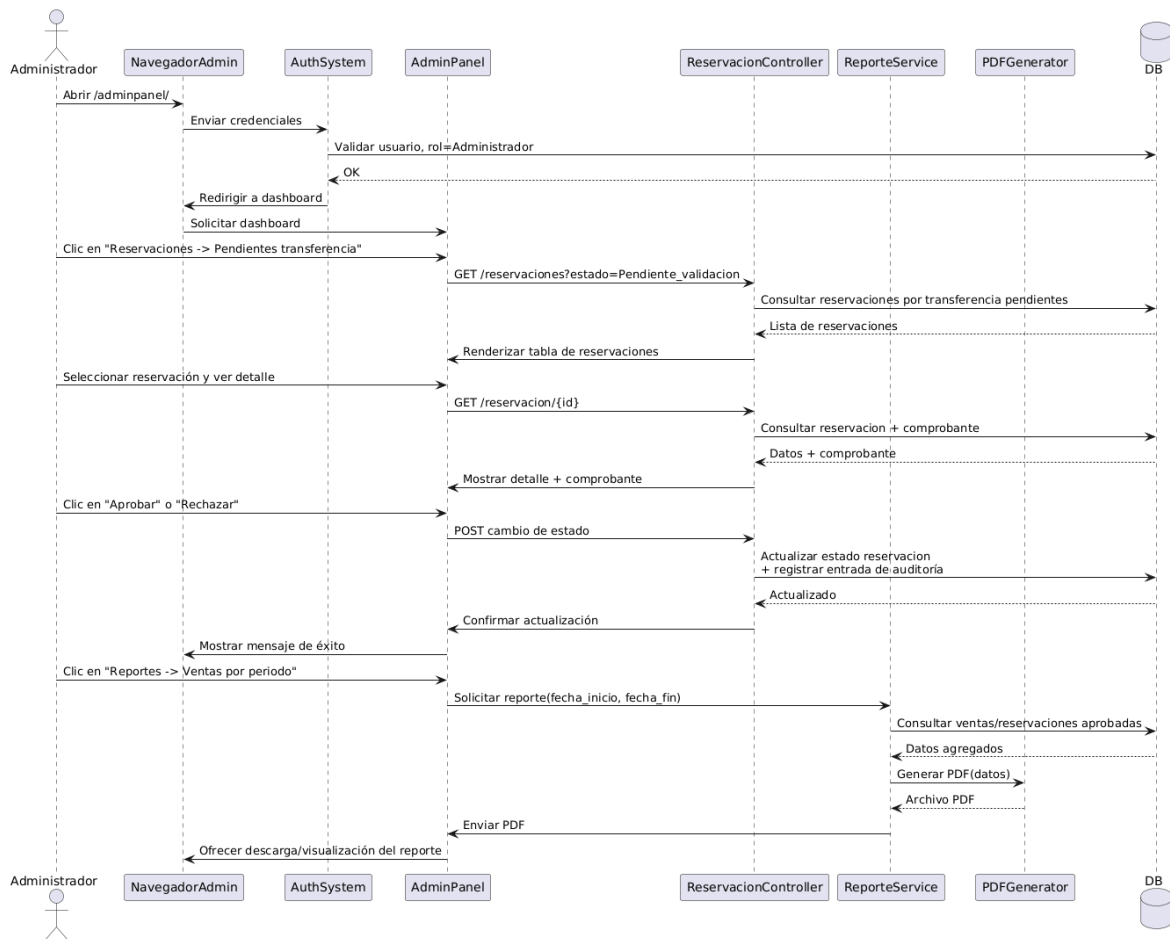


Figura 12 Diagrama de secuencia - Administrador

Escenario: validar pago por transferencia y generar reporte de ventas

Según tu documento:

- El **Administrador** tiene acceso a todas las funciones del panel, incluyendo reportes y métricas.
- Puede **modificar estados de reservaciones** y **validar comprobantes**.
- Puede generar **reportes en PDF** y ver métricas.

Participantes

- Administrador
- NavegadorAdmin
- AuthSystem
- AdminPanel

- ReservacionController
- ReporteService (servicio para generar reportes)
- PDFGenerator
- DB

Flujo narrado

1. El **Administrador** inicia sesión en /adminpanel/.
2. AuthSystem valida credenciales + rol Administrador.
3. AdminPanel muestra el **dashboard** con indicadores clave.
4. El Administrador entra a “**Reservaciones** → **Pendientes por transferencia**”.
5. ReservacionController consulta en la **DB** las reservaciones con método Transferencia y estado Pendiente validación.
6. El Administrador elige una reservación y abre su **detalle**.
7. ReservacionController recupera el comprobante de la **DB** y lo envía al AdminPanel (para verlo/descargarlo).
8. Tras revisar el comprobante, el Administrador selecciona “**Aprobar**” o “**Rechazar**”.
9. ReservacionController actualiza en la **DB** el estado de la reservación y registra la acción (auditoría/novedad).
10. El Administrador después entra a “**Reportes** → **Ventas por periodo**”.
11. AdminPanel envía la solicitud a ReporteService con el rango de fechas.
12. ReporteService consulta la **DB** (ventas/reservaciones aprobadas) y arma los datos del reporte.
13. ReporteService llama a PDFGenerator para construir el PDF.
14. PDFGenerator devuelve el archivo al ReporteService, que lo envía al AdminPanel.
15. El **AdminPanel** ofrece el PDF para descarga o visualización al Administrador.

3.5.7.3. Diagrama de secuencia – Desarrollador

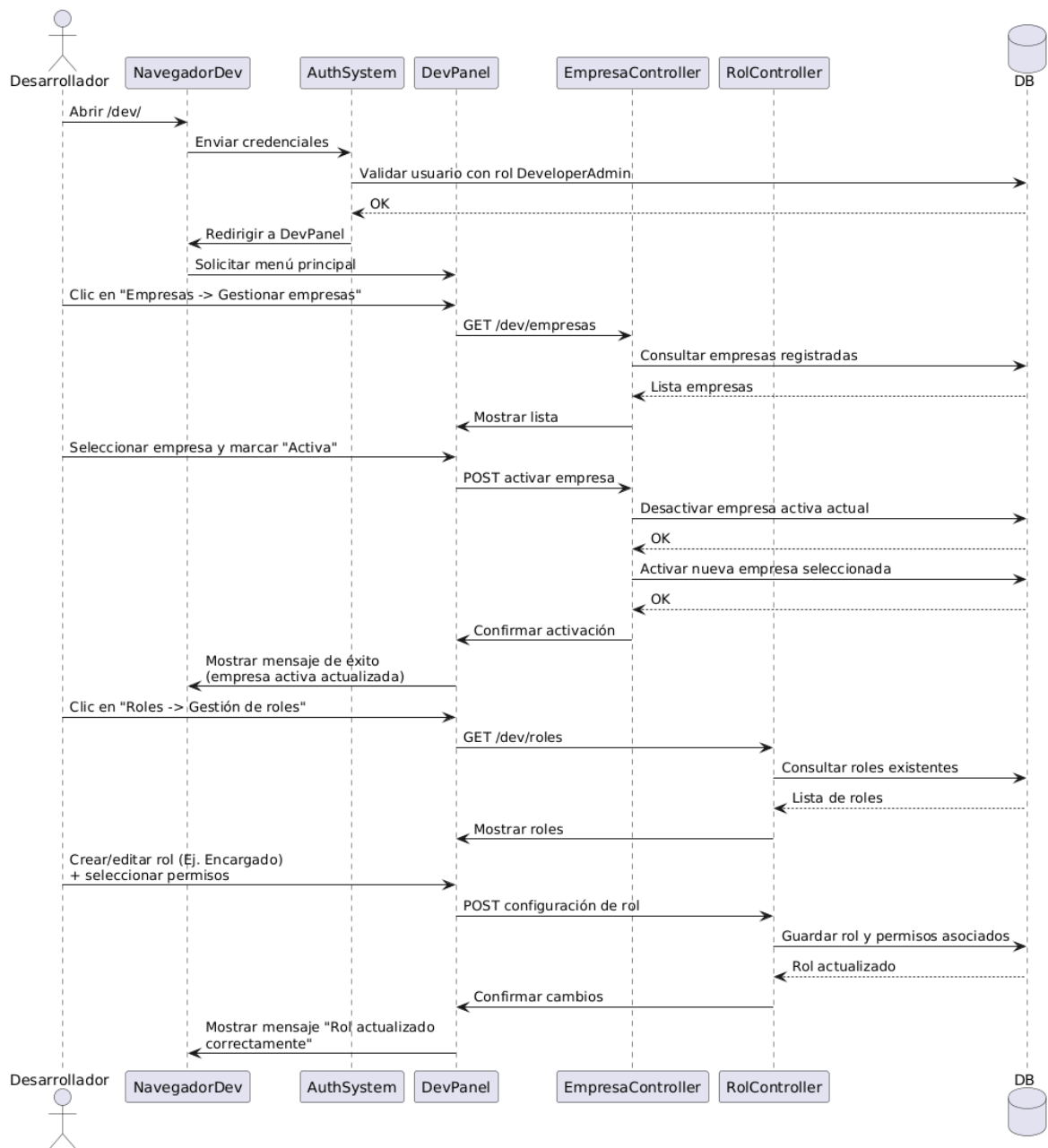


Figura 13 Diagrama de secuencia - Desarrollador

Escenario: activar una empresa y configurar roles en el devpanel

Según el documento:

- El **Panel de Desarrollador** es el nivel más alto de administración.
- El Desarrollador puede:
 - Registrar y activar empresas.

- Configurar tema visual.
- Crear roles personalizados y asignar permisos.

Participantes

- Desarrollador
- NavegadorDev
- AuthSystem
- DevPanel (aplicación devpanel/)
- EmpresaController
- RolController
- DB

Flujo narrado

Parte A: Activar empresa

1. El **Desarrollador** entra a /dev/.
2. AuthSystem valida que sea un usuario con rol **DeveloperAdmin** / superusuario.
3. DevPanel muestra el menú principal.
4. El Desarrollador va a “**Empresas → Gestionar empresas**”.
5. EmpresaController consulta en la **DB** las empresas registradas y muestra la lista.
6. El Desarrollador selecciona una empresa y la marca como “**Activa**”.
7. EmpresaController actualiza la **DB**:
 - Desactiva la empresa activa anterior.
 - Marca la nueva empresa como activa = True.
8. El DevPanel actualiza el contexto global (context processors) para que el sitio público use los datos de la nueva empresa.

Parte B: Configurar roles y permisos

9. El Desarrollador accede a “**Roles → Gestión de roles**”.
10. RolController consulta la **DB** y muestra roles existentes (DeveloperAdmin, CompanyAdmin, Empleado, Encargado, Cliente, etc.).
11. El Desarrollador crea o edita un rol (por ejemplo, Encargado) asignando permisos:

- Gestionar reservaciones.
- Validar pagos por transferencia.
- Acceder al panel administrativo.

12. RolController guarda la configuración en la **DB**.

13. El sistema queda listo para que los usuarios internos usen esos roles en el adminpanel.

3.5.8. Diagrama de actividades

El diagrama de actividades representa el flujo de operaciones que realizan los diferentes actores dentro del sistema, desde el momento en que un visitante interactúa con el sitio hasta la gestión interna por parte de los administradores.

El modelo está dividido en cuatro carriles principales que delimitan las responsabilidades de cada tipo de usuario: **Visitante**, **Usuario registrado**, **Administrador del negocio** y **Administrador del sitio**.

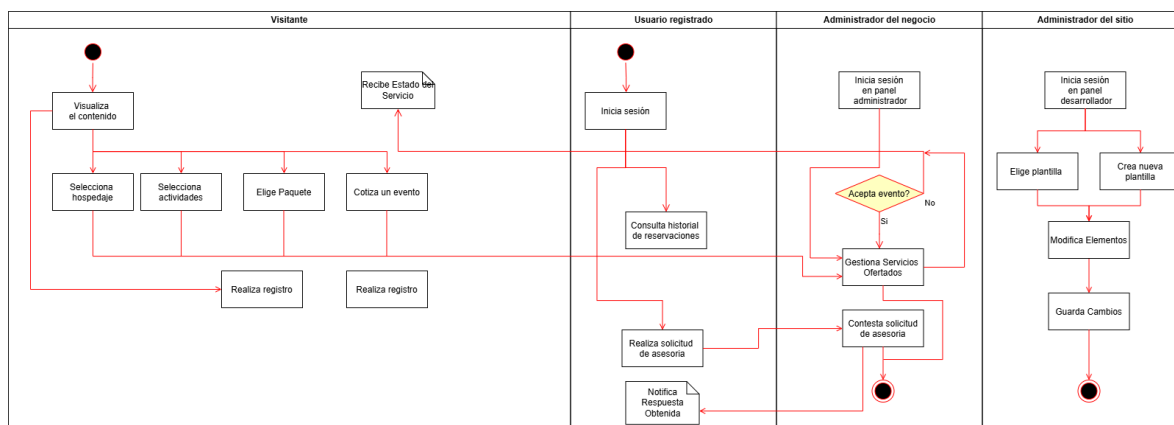


Figura 14 Diagrama de actividades

1. Administrador del negocio

- Este rol tiene acceso al panel administrativo, donde gestiona los servicios ofrecidos y supervisa las solicitudes entrantes.
- Cuando se recibe una solicitud de evento o asesoría, el administrador evalúa si acepta o no el evento.
 - Si la respuesta es afirmativa, gestiona los servicios ofertados, ajustando disponibilidad, precios o características.

- c. Finalmente, contesta la solicitud de asesoría, generando la notificación que llega al usuario registrado.

2. Administrador del sitio (desarrollador)

- a. Inicia sesión en el panel de desarrollador.
- b. Puede elegir una plantilla existente o crear una nueva para personalizar la interfaz del sitio.
- c. Posteriormente modifica los elementos visuales o estructurales del sistema y guarda los cambios, manteniendo la consistencia del entorno visual y funcional.

Síntesis del flujo

Este diagrama evidencia la coordinación entre los distintos actores del sistema:

- Los **visitantes y usuarios registrados** representan la capa de interacción externa.
- El **administrador del negocio** actúa como intermediario entre las solicitudes de los clientes y los servicios disponibles.
- El **administrador del sitio** mantiene la infraestructura y personalización del sistema.

El flujo asegura la continuidad operativa del sistema desde la navegación inicial hasta la administración interna, ilustrando claramente los puntos de contacto entre los diferentes roles.

3.5.9. Diagrama de despliegue

El diagrama de despliegue muestra la distribución física del sistema, es decir, cómo los componentes del software se instalan y comunican en la infraestructura tecnológica.

Este diagrama resulta fundamental para comprender la arquitectura cliente-servidor implementada en la aplicación.

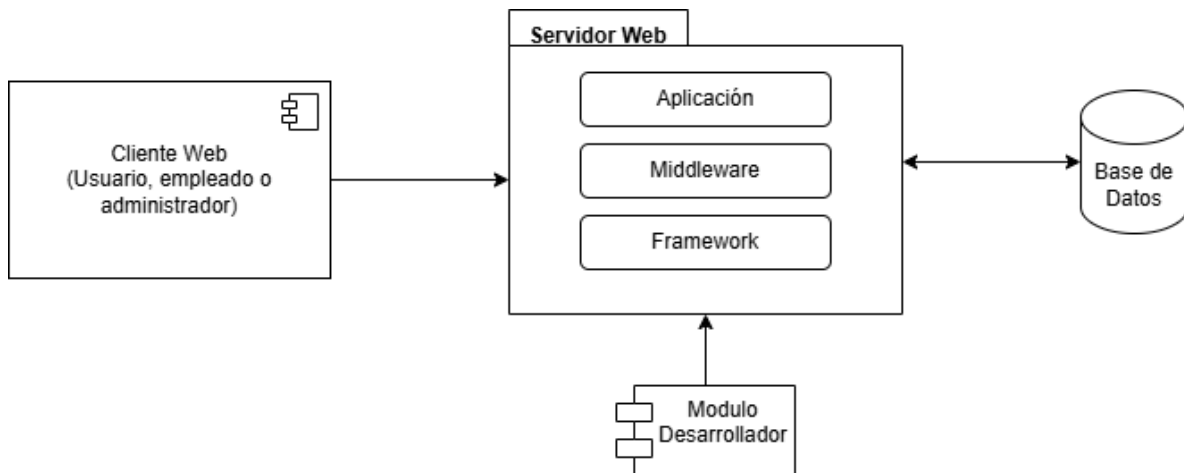


Figura 15 Diagrama de despliegue

Descripción:

El diagrama de despliegue está conformado por los siguientes nodos:

- **Nodo Cliente Web:** representa el dispositivo del usuario (computadora, tableta o teléfono) que accede al sistema a través de un navegador.
- **Nodo Servidor de Aplicaciones:** aloja el proyecto Django, maneja la lógica de negocio, las rutas y los controladores. Está configurado con Gunicorn o Apache y conectado al servidor de base de datos.
- **Nodo Servidor de Base de Datos:** contiene las tablas del sistema y responde a las consultas SQL ejecutadas desde el backend.
- **Nodo Servicio PayPal:** representa la pasarela de pagos externa, encargada de procesar las transacciones seguras.

El flujo de comunicación se realiza mediante el protocolo HTTPS, garantizando la seguridad en la transferencia de datos entre el cliente, el servidor y los servicios externos.

Esta arquitectura distribuida asegura un rendimiento estable, escalabilidad y disponibilidad permanente.

3.5.10. Conclusión del modelado

El modelado del sistema permitió definir con claridad la estructura lógica, la organización de los datos, los flujos de información y la arquitectura técnica del

software.

Gracias a los diagramas UML elaborados, se obtuvo una visión integral de la aplicación, lo que facilitó la transición hacia la etapa de desarrollo e implementación.

Los modelos de clases y entidad–relación garantizaron una base sólida para el manejo de datos; los diagramas de secuencia evidenciaron la comunicación entre los componentes; y los diagramas de componentes y despliegue confirmaron la viabilidad técnica y modular del sistema. En conjunto, estos elementos aseguran que la solución propuesta sea robusta, escalable y alineada con las buenas prácticas de la ingeniería de software moderna.

3.6. Diseño de la Base de Datos

El diseño de la base de datos representa una de las etapas más importantes dentro del desarrollo del sistema, pues define la forma en que la información será almacenada, relacionada y consultada a lo largo de todo el ciclo de vida de la aplicación.

En este proyecto, se implementó una base de datos relacional administrada mediante MySQL, aprovechando su robustez, rendimiento y compatibilidad con el ORM (Object Relational Mapping) del framework Django. El ORM permite mantener una capa de abstracción entre los modelos de datos definidos en el código y las tablas reales de la base de datos, garantizando consistencia y portabilidad.

Tabla 15 Tablas en la base de datos

Tabla	Descripción	Clave primaria	Relaciones principales
Empresa	Almacena la información general de cada empresa, su configuración y datos de contacto.	id	1–N con <i>Usuario</i> , <i>Rol</i> , <i>Servicio</i> , <i>Producto</i> , <i>Pedido</i> , <i>Reservación</i> , <i>Tema</i> , <i>Página</i> .
Tema	Define los esquemas visuales y colores personalizados para cada empresa.	id	1–N con <i>Empresa</i> (asignado como FK en Empresa).
CustomUser	Representa a los usuarios del sistema con roles	id	N–1 con <i>Empresa</i> , N–1 con <i>Rol</i> , 1–N con

	personalizados y datos de contacto.		<i>Pedido</i> , 1–N con <i>Reseña</i> .
Rol	Contiene los roles dinámicos definidos por cada empresa (Administrador, Empleado, etc.).	id	1–N con <i>CustomUser</i> (campo rol), N–1 con <i>Empresa</i> .
Página	Define las páginas principales del sitio (Home, Servicios, Productos, etc.) para cada empresa.	id	N–1 con <i>Empresa</i> , 1–N con <i>Sección</i> .
Sección	Subdivisiones internas de las páginas (hero, texto, carrusel, testimonios, etc.).	id	N–1 con <i>Página</i> .
TipoServicio	Clasifica los servicios según tipo de contratación (por día, por hora, etc.).	id	N–1 con <i>Empresa</i> , 1–N con <i>Servicio</i> .
Servicio	Registra la información de cada servicio ofertado (nombre, costo, descripción, imagen).	id	N–1 con <i>TipoServicio</i> , N–1 con <i>Empresa</i> , 1–N con <i>ImagenServicio</i> , 1–N con <i>Reseña</i> , N–N con <i>Reservación</i> .
ImagenServicio	Contiene las imágenes adicionales asociadas a un servicio.	id	N–1 con <i>Servicio</i> .
Reseña	Reseñas y calificaciones generadas por los usuarios sobre servicios o productos.	id	N–1 con <i>CustomUser</i> , N–1 con <i>Servicio</i> o <i>Producto</i> (relación polimórfica).
Reservación	Información sobre las reservaciones realizadas (cliente, fechas, estado, pago).	id	N–1 con <i>Empresa</i> , 1–N con <i>Reservacion_servicio</i> .
Reservacion_servicio	Asociación entre reservaciones y servicios contratados.	id	N–1 con <i>Reservación</i> , N–1 con <i>Servicio</i> .
Producto	Define los productos que una empresa ofrece para venta directa o catálogo.	id	N–1 con <i>Empresa</i> , N–1 con <i>CategoríaProducto</i> , 1–N con <i>ItemPedido</i> , 1–N con <i>Reseña</i> .
CategoríaProducto	Clasificación de productos (ej.	id	N–1 con <i>Empresa</i> , 1–N con <i>Producto</i> .

	alimentos, souvenirs, accesorios).		
Pedido	Encabezado de un pedido o compra, con información de cliente y método de pago.	id	N–1 con <i>Empresa</i> , N–1 con <i>CustomUser</i> , 1–N con <i>ItemPedido</i> .
ItemPedido	Registra los productos individuales dentro de un pedido.	id	N–1 con <i>Pedido</i> , N–1 con <i>Producto</i> .
Venta	Control de las ventas registradas por la empresa (monto, fecha, estado).	id	N/A — Independiente, sin FK directa (opcionalmente relacionada con <i>Empresa</i>).
Novedad	Bitácora de acciones relevantes dentro del sistema.	id	N–1 con <i>CustomUser</i> .
ContactMessage	Mensajes enviados desde el formulario de contacto del sitio.	id	N–1 con <i>Empresa</i> .
CustomAttribute	Define atributos personalizados para modelos como <i>Producto</i> o <i>Servicio</i> .	id	N–1 con <i>Empresa</i> .

El propósito principal del diseño fue construir una estructura modular, escalable y multiempresa, capaz de soportar de manera independiente la información de diferentes organizaciones dentro del mismo sistema, sin duplicar la lógica ni comprometer la integridad de los datos.

3.6.1. Arquitectura general del modelo de datos

La base de datos se compone de un conjunto de entidades interrelacionadas que representan los distintos módulos funcionales del sistema: autenticación, empresas, servicios, reservaciones, productos, pedidos, ventas, personalización y comunicación.

Cada módulo fue diseñado con un alto grado de cohesión interna y bajo acoplamiento externo, siguiendo las buenas prácticas de normalización (3FN) para evitar redundancias y facilitar el mantenimiento.

El modelo se basa en una jerarquía de dependencias centrada en la entidad Empresa, la cual actúa como nodo raíz del sistema. A partir de esta, se derivan los diferentes submódulos (usuarios, servicios, productos, configuraciones, páginas, etc.), asegurando que cada empresa gestione de forma aislada su información.

A nivel conceptual, el diseño general sigue la siguiente lógica:

1. **Empresa:** núcleo de identidad, configuración y personalización.
2. **Usuarios y roles:** administración de accesos y privilegios.
3. **Servicios y productos:** catálogo de oferta comercial.
4. **Reservaciones y pedidos:** control de operaciones y ventas.
5. **Interacción y comunicación:** recepción de mensajes y reseñas.
6. **Configuración visual:** manejo de páginas, secciones y temas.
7. **Auditoría:** registro de actividades internas.

Esta estructura modular permite mantener la escalabilidad del sistema, facilitando la incorporación futura de nuevos módulos sin alterar la estructura base.

3.6.2. Descripción detallada de las entidades

A continuación, se describen las entidades más relevantes del modelo, agrupadas por módulo funcional:

Módulo de Empresas y Configuración

- **Empresa:** Entidad principal del sistema. Contiene la información institucional (nombre, contacto, titular, ubicación), datos bancarios (cuenta, CLABE, tarjeta), parámetros de personalización visual (logotipo, dominio, subdominio) y configuración de correo (SMTP). Incluye campos de control (activa, fecha_creacion) y banderas que habilitan módulos específicos (productos_habilitado, servicios_habilitado, resenas_habilitado). Su relación con **Tema** y con todos los demás módulos (usuarios, servicios,

productos, páginas, etc.) permite el funcionamiento multiempresa del sistema.

- **Tema:** Define la apariencia visual del sitio web de cada empresa, almacenando los colores primario, secundario y de acento, fuentes tipográficas, estilo de navbar e imágenes como logotipo o hero principal. Esta tabla se relaciona 1:N con Empresa, permitiendo que varias empresas puedan compartir o reutilizar un mismo tema base.

Módulo de Usuarios y Roles

- **CustomUser:** Extiende el modelo AbstractUser de Django para integrar campos adicionales como nombre_completo, fecha_nacimiento, telefono y empresa. Se le asocia un campo rol, que hace referencia al modelo Rol, y un campo atributos_personalizados en formato JSON que permite almacenar valores definidos por la empresa según el rol asignado. Este enfoque ofrece una gran flexibilidad para gestionar distintos tipos de personal sin modificar la estructura del modelo.
- **Rol:** Representa los roles dinámicos definidos por cada empresa, como “Administrador”, “Empleado” o “Encargado”. Contiene un esquema JSON llamado atributos_schema, donde se definen los campos personalizados que tendrán los usuarios con ese rol, junto con los permisos específicos. La relación Empresa–Rol–Usuario forma un triángulo lógico que facilita la administración descentralizada.

Módulo de Servicios y Reseñas

- **TipoServicio:** Permite clasificar los servicios de una empresa en diferentes modalidades, como “por hora” o “por día”. Está vinculado directamente a Empresa, lo que asegura que cada organización defina sus propios tipos de servicios.

- **Servicio:** Representa la unidad principal de oferta comercial. Contiene campos como título, descripción, precios (costo_por_persona, costo_niño, costo_con_descuento), imagen principal, duración y restricciones. Su campo atributos_personalizados permite agregar detalles únicos por empresa sin modificar la estructura general. Se relaciona con TipoServicio y Empresa, lo que garantiza consistencia jerárquica.
- **ImagenServicio:** Almacena imágenes adicionales asociadas a un servicio específico. Se utiliza para construir galerías visuales dinámicas en el sitio web. Incluye orden y descripción, con relación N:1 hacia Servicio.
- **Resena:** Permite a los clientes calificar y comentar sobre servicios o productos. Utiliza el sistema de ContentType de Django para relacionarse genéricamente con diferentes tipos de objetos, evitando la duplicación de modelos. Incluye control de aprobación y fechas de creación y actualización.

Módulo de Reservaciones

- **Reservacion:** Gestiona las solicitudes de reserva realizadas por los clientes, incluyendo datos personales, fechas, número de personas, comentarios, estado y comprobante de pago. Se vincula con Empresa para asegurar la segmentación de datos y con Reservacion_servicio para registrar los servicios reservados. Los estados disponibles (pendiente, aprobada, finalizada) permiten el seguimiento del ciclo de vida de la reserva.
- **Reservacion_servicio:** Actúa como una tabla intermedia que relaciona cada reservación con los servicios contratados, configurando una relación N:M lógica. Incluye una propiedad que permite acceder fácilmente a la empresa asociada.

Módulo de Productos, Ventas y E-Commerce

- **CategoriaProducto:** Clasifica los productos dentro de una empresa. Los campos principales son nombre, descripcion y empresa_id. Mantiene integridad referencial y unicidad por empresa.
- **Producto:** Almacena la información detallada de cada artículo del catálogo: nombre, descripción, precio, stock, SKU, imagen y fecha de creación. También utiliza un campo atributos_personalizados (JSON) para extender dinámicamente la descripción del producto. Se relaciona con Empresa y CategoriaProducto.
- **Pedido:** Registra las órdenes de compra realizadas por los clientes. Incluye datos del cliente (nombre_cliente, direccion_entrega), método de pago, estado, total y fechas de creación y entrega estimada. Contiene una relación 1:N con ItemPedido, lo que permite vincular múltiples productos a un solo pedido. También se relaciona con Usuario (si la compra fue realizada por un cliente autenticado) y Empresa.
- **ItemPedido:** Representa los productos específicos dentro de un pedido, almacenando el nombre, precio y cantidad al momento de la transacción. Su inclusión garantiza que las variaciones futuras en los precios no alteren los registros históricos de ventas.
- **Venta:** Entidad utilizada para registrar transacciones individuales o ventas directas. Incluye monto, fecha y estado (pendiente, pagado, cancelado).
- **Novedad:** Actúa como una bitácora de eventos internos. Registra el usuario, la acción realizada y la fecha de ejecución, brindando trazabilidad y control administrativo.

Módulo de Comunicación y Soporte

- **ContactMessage:** Registra los mensajes enviados desde el formulario de contacto de cada empresa. Almacena los datos dinámicos (nombre, correo, mensaje) en formato JSON, junto con metadatos como dirección IP, agente

de usuario y estado de lectura o respuesta. Este modelo centraliza la comunicación entre los clientes y la empresa.

Módulo de Personalización Visual y Contenido

- **Pagina:** Representa cada una de las páginas del sitio web de una empresa (Inicio, Servicios, Productos, Contacto, etc.). Incluye el campo slug para identificarla, un título, estado (activa) y fechas de creación y modificación. Se relaciona con Empresa y garantiza unicidad combinada (empresa, slug).
- **Seccion:** Define los bloques o componentes dentro de una página. Cada sección puede tener un tipo (hero, carousel, features, mapa, testimonios, etc.) y un campo configuracion (JSON) donde se almacenan textos, imágenes y parámetros específicos. Esto otorga gran flexibilidad para construir sitios dinámicos sin alterar la estructura de la base de datos.
- **CustomAttribute:** Permite definir atributos personalizados para modelos específicos (por ejemplo, agregar “Talla” a productos o “Duración” a servicios).
Se relaciona con Empresa y mantiene unicidad por combinación (empresa, target_model, name), lo que evita duplicados y asegura consistencia.

3.6.3. Relaciones entre entidades

El modelo mantiene relaciones jerárquicas cuidadosamente diseñadas:

- **Empresa → Usuario → Rol:** cada empresa puede tener múltiples usuarios, y cada usuario tiene un rol que define sus permisos y atributos personalizados.
- **Empresa → Servicio → Reservacion:** los servicios pertenecen a una empresa y pueden ser reservados varias veces.
- **Empresa → Producto → Pedido → ItemPedido:** las empresas gestionan sus productos y los pedidos asociados.

- **Empresa → Pagina → Seccion:** cada empresa puede estructurar su sitio web a medida mediante páginas y secciones.
- **Empresa → ContactMessage:** centraliza los mensajes recibidos del público.
- **Usuario → Novedad:** documenta las acciones de administración y mantenimiento del sistema.

El uso de relaciones foráneas (ForeignKey) con reglas de eliminación controladas (CASCADE, SET_NULL, PROTECT) garantiza la integridad referencial de los datos, evitando registros huérfanos o inconsistentes.

3.6.4. Normalización, integridad y optimización

El modelo de base de datos se encuentra **normalizado hasta la tercera forma normal (3FN)**, lo que asegura:

- **Ausencia de redundancia:** cada dato se almacena en una sola tabla.
- **Integridad referencial:** las relaciones entre entidades están claramente definidas mediante claves primarias y foráneas.
- **Eficiencia en consultas:** las búsquedas y filtros se optimizan gracias a índices automáticos en campos clave.
- **Flexibilidad controlada:** el uso de JSONField permite ampliar datos sin romper la estructura normalizada.

Además, se implementaron estrategias adicionales de optimización:

- Definición de índices en campos de búsqueda frecuentes (email, slug, sku).
- Carga diferida de relaciones (select_related, prefetch_related) a nivel ORM.
- Control de integridad mediante validadores (RegexValidator, unique_together).
- Auditoría continua de cambios a través del modelo Novedad.

3.6.5. Escalabilidad y modularidad

El diseño permite escalar horizontalmente al incorporar nuevas empresas, usuarios o módulos sin afectar el rendimiento. Cada empresa opera de forma independiente dentro del mismo esquema lógico, manteniendo sus propios registros y configuraciones.

La modularidad del sistema posibilita agregar futuras funcionalidades —como facturación electrónica, reportes estadísticos o módulos de soporte— sin alterar las tablas existentes. El uso del ORM de Django facilita las migraciones automáticas y garantiza compatibilidad hacia distintos motores de base de datos (MySQL, PostgreSQL, SQLite).

3.6.6. Conclusión del diseño de base de datos

El modelo diseñado cumple con los principios fundamentales de la ingeniería de software: coherencia, integridad, eficiencia y escalabilidad. Gracias a la adecuada aplicación de técnicas de modelado relacional y la integración con el ORM de Django, la base de datos garantiza una gestión segura y flexible de la información. Cada entidad cumple un propósito claro dentro del flujo operativo, y su interconexión permite un manejo integral de empresas, servicios, productos, usuarios y transacciones.

En conjunto, el diseño de la base de datos constituye la columna vertebral del sistema, sobre la cual se construyen los módulos de negocio y la interfaz web, asegurando un rendimiento óptimo y una administración eficiente de los recursos de información.

3.7. Diseño de la Interfaz

El diseño de la interfaz gráfica constituye una de las fases más relevantes dentro del proceso de desarrollo del sistema, ya que determina la manera en que los usuarios interactúan con las funcionalidades disponibles. Para este proyecto, se aplicaron principios de Diseño Centrado en el Usuario (UCD) y UX/UI Design, con el objetivo de lograr una experiencia visual coherente, intuitiva

y eficiente para los distintos perfiles de usuario (cliente, administrador y desarrollador).

El diseño de la interfaz se realizó mediante la herramienta Figma, la cual permitió representar de manera clara y funcional las pantallas del sistema antes de su implementación definitiva.

El enfoque seguido busca garantizar consistencia visual, facilidad de uso y una navegación fluida, asegurando que cada tipo de usuario pueda acceder de forma intuitiva a las funciones que le corresponden.

Los mockups se dividieron en tres apartados que corresponden a los módulos principales del sistema: el lado cliente, el panel administrativo y el panel desarrollador.

3.7.1. Panel Administrativo

Corresponde a la interfaz utilizada por los administradores y encargados de negocio. Su diseño busca ofrecer una vista ordenada y funcional para la gestión de los datos operativos.

Mockups elaborados:

- Dashboard principal: presenta indicadores clave (ventas, servicios activos, usuarios).
- Gestión de servicios y productos: permite agregar, editar o desactivar elementos del catálogo.
- Módulo de reservaciones: muestra las solicitudes pendientes y aprobadas, con opción de cambiar su estado.
- Gestión de usuarios y roles: posibilita la administración de accesos, permisos y datos de contacto.
- Reportes y novedades: muestra información resumida del sistema y un registro de eventos relevantes.

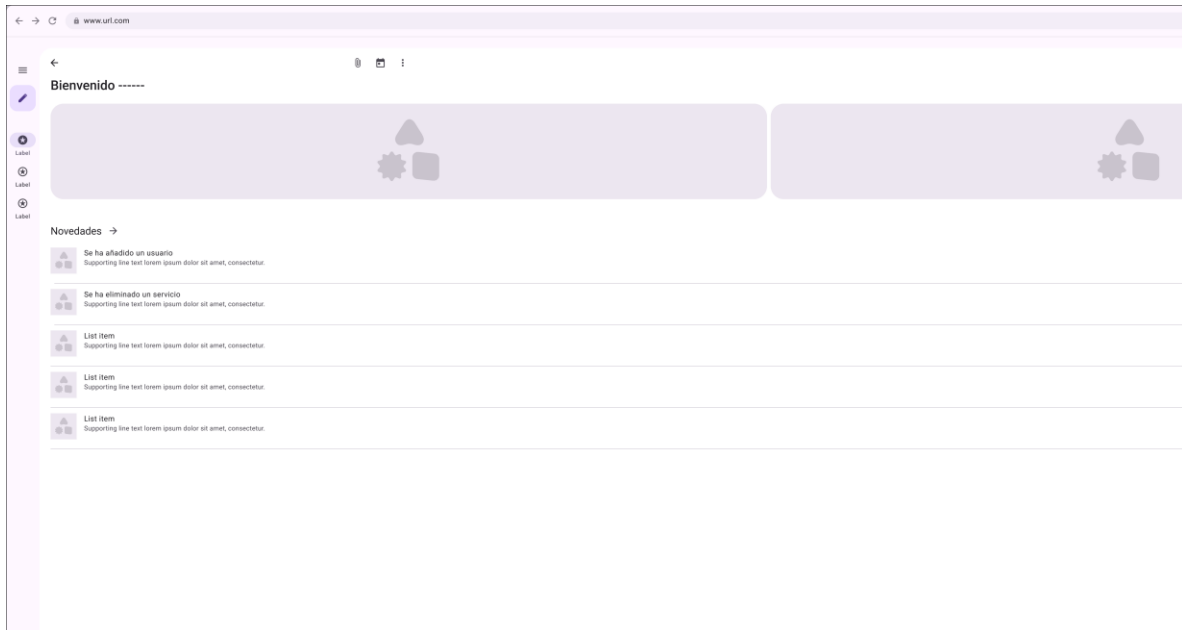


Figura 16 Dashboard

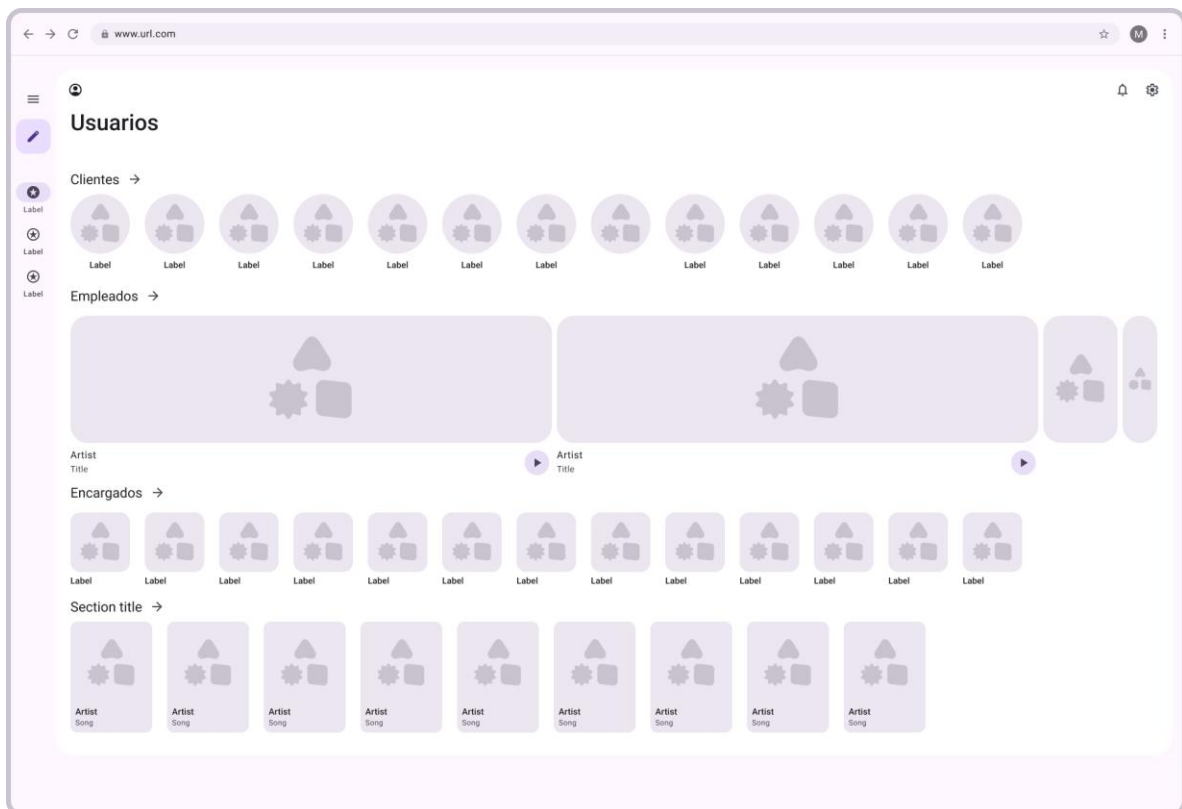


Figura 17 Panel de usuarios

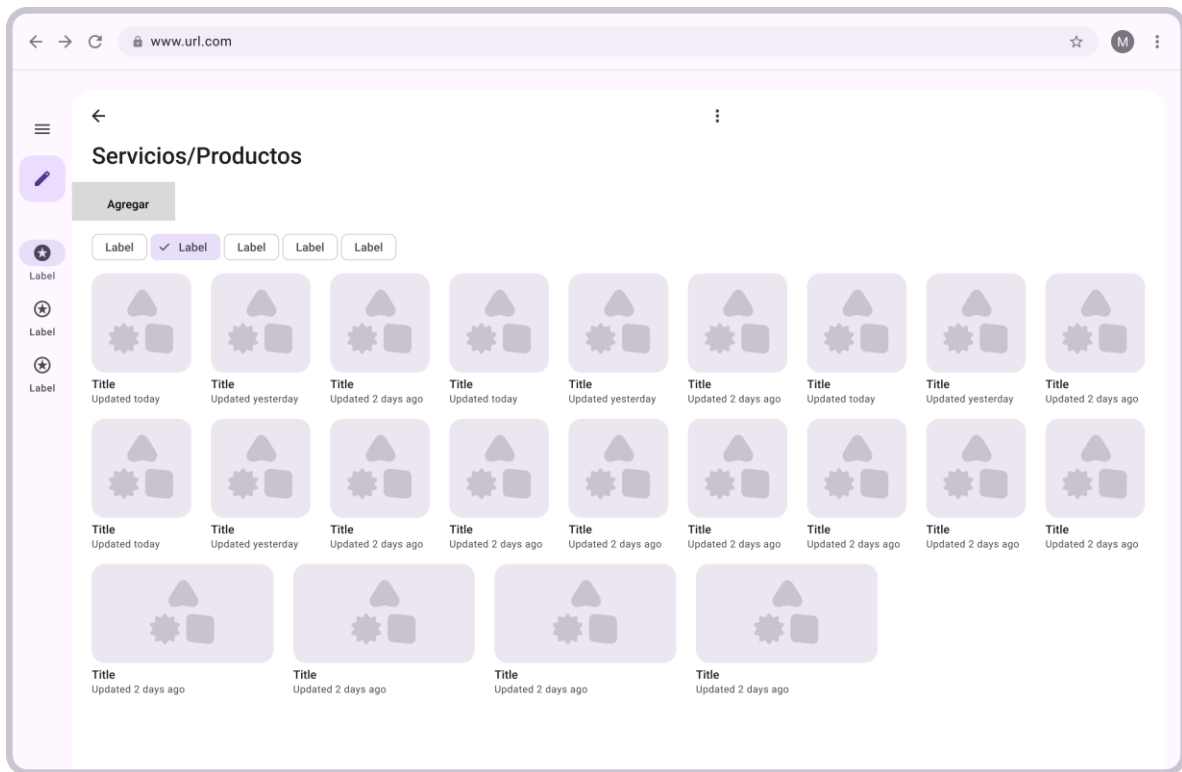


Figura 18 Productos/Servicios

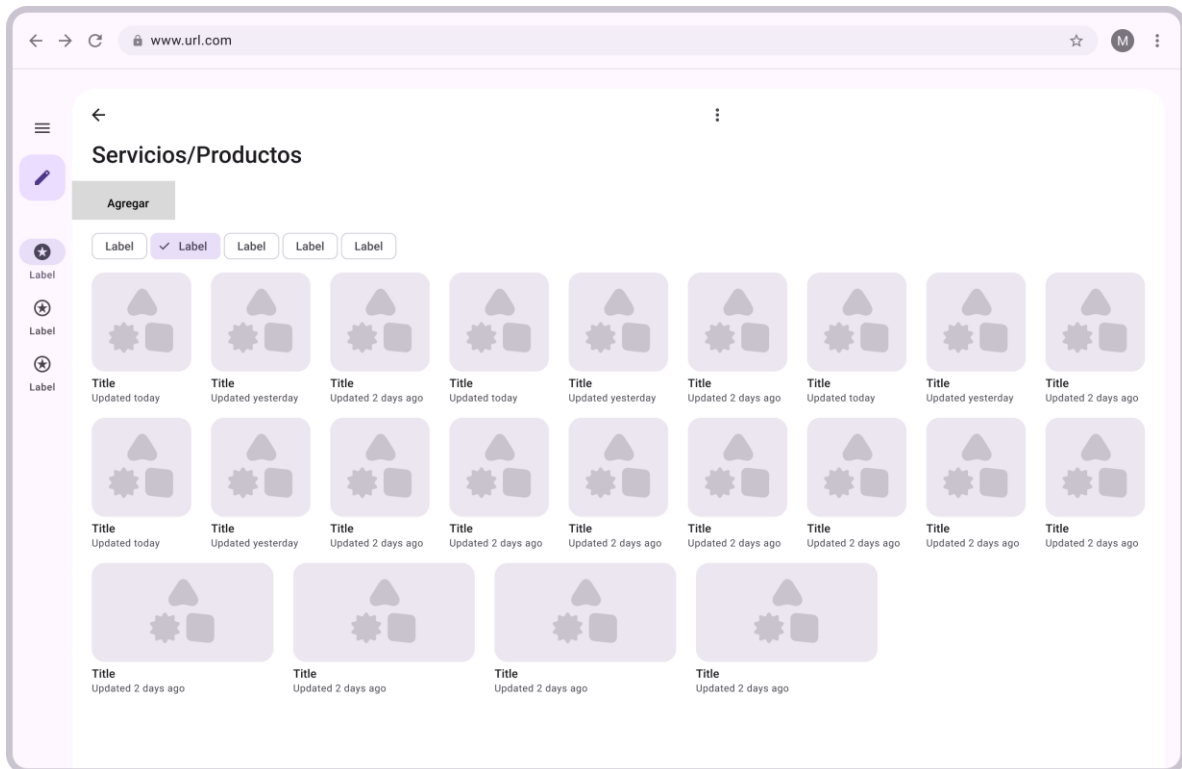


Figura 19 Detalles

Heading
Subheading



Title

Body text for whatever you'd like to say. Add main takeaway points, quotes, anecdotes, or even a very very short story.

Button



Title

Body text for whatever you'd like to say. Add main takeaway points, quotes, anecdotes, or even a very very short story.

Button



Title

Body text for whatever you'd like to say. Add main takeaway points, quotes, anecdotes, or even a very very short story.

Button

Figura 20 Listado de reservaciones

Sales					Contacts					Deals					Tasks					Appointments					Mail					Calendar				
Deals (41)					Sort by Created Date					Your sales track Default					IMPORT					Actions					Create Now									
New 22 >>>					Prospect 4 >>>					Proposal 10 >>>					Won 3 >>>					Lost 2 >>>														
Andy (\$150.25) Close Date: 11/22/2018					Mike (\$500) Close Date: 10/09/2018					Sharon (\$100) Close Date: --					Jack (\$350) Close Date: 10/07/2018					Ashley (\$300) Close Date: 10/08/2018														
Sharon (\$232) Close Date: --					Anna (\$250) Close Date: 01/08/2019					Peter (\$100) Close Date: --					Stanley (\$250) Close Date: 10/08/2018					Kevin (\$400) Close Date: 11/08/2018														
Eric (\$232) Close Date: --					David (\$400) Close Date: 08/08/2018					Roy (\$100) Close Date: --					Brian (\$400) Close Date: 02/08/2019																			
John (\$250)					Alex (\$300)					Sarah (\$280)																								
Total: \$8508.50					Total: \$14500.00					Total: \$6880.00					Total: \$1000.00					Total: \$700.00														

Figura 21 Apartado de ventas

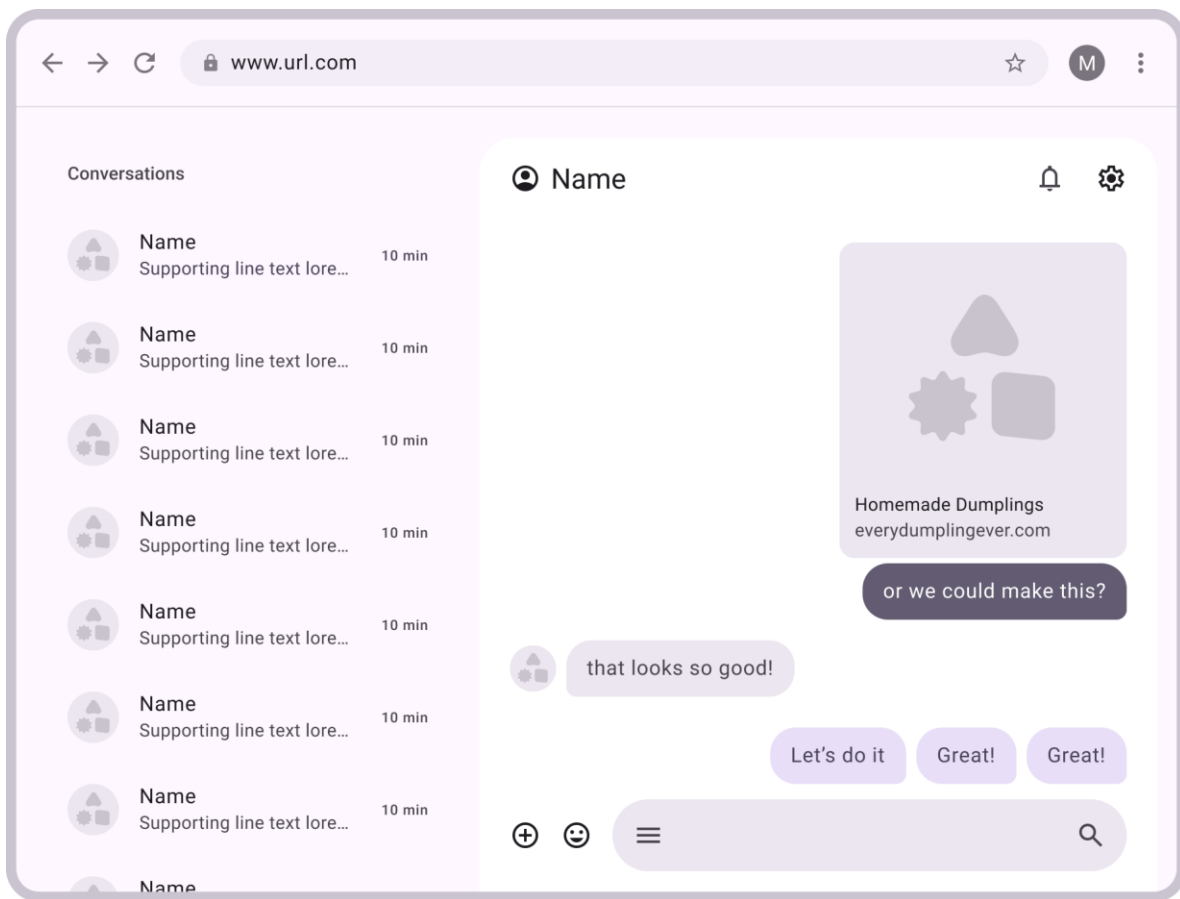


Figura 22 Chat interno

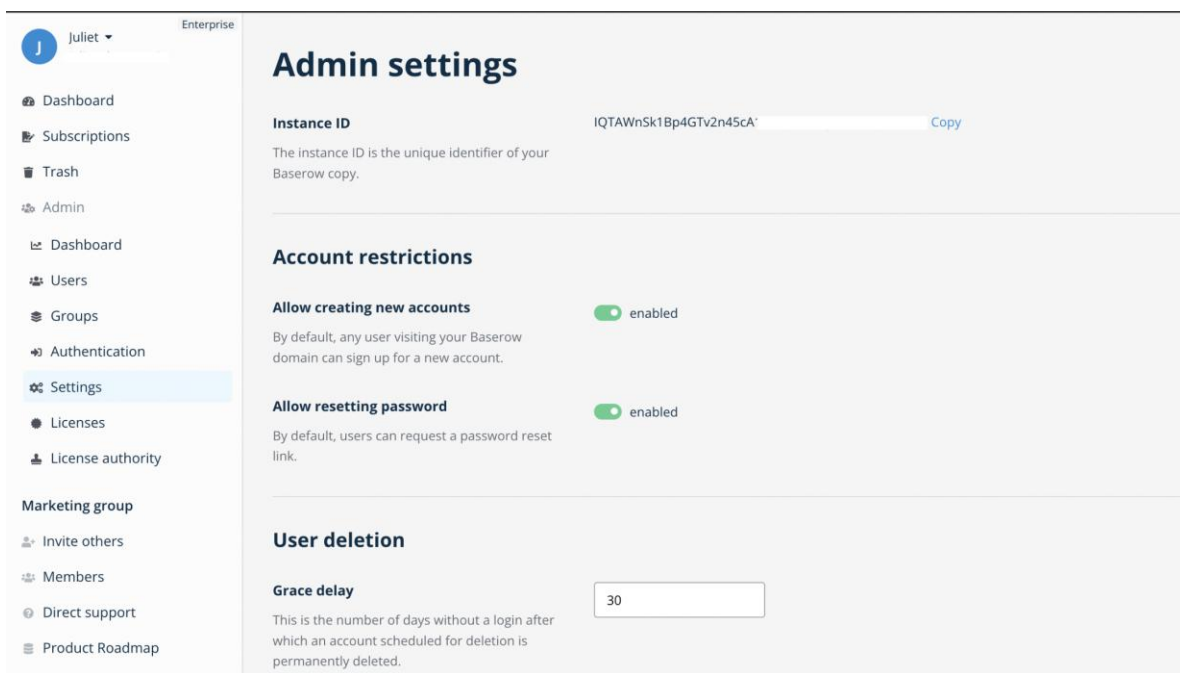


Figura 23 Panel de configuración

El diseño se centró en la usabilidad, la consistencia entre pantallas y la organización por prioridad funcional, utilizando componentes visuales simples y una estructura de panel lateral con navegación vertical.

3.7.2. Panel del Desarrollador

Esta parte del sistema está orientada al personal técnico o de mantenimiento, responsable de la personalización visual y funcional de los sitios asociados.

Mockups generados:

- Pantalla de login del desarrollador.
- Panel de gestión de temas: permite elegir o crear plantillas personalizadas.
- Editor de componentes: interfaz para modificar elementos visuales (colores, fuentes, estructuras).
- Vista previa y guardado: permite visualizar los cambios aplicados y almacenarlos en el sistema.

DevPanel

Gestion Operativa Web

Welcome back!

Email address

example@mail.com

Password

Password



Use at least 8 characters with 1 number, and one special character.

LOG IN

[Forgot password?](#)

Figura 24 Login

DevPanel

Gestion Operativa Web

Welcome back!

Email address

example@mail.com



Password

••••••••



Use at least 8 characters with 1 number, and one special character.

LOG IN

[Forgot password?](#)

Figura 35 Login(activo)

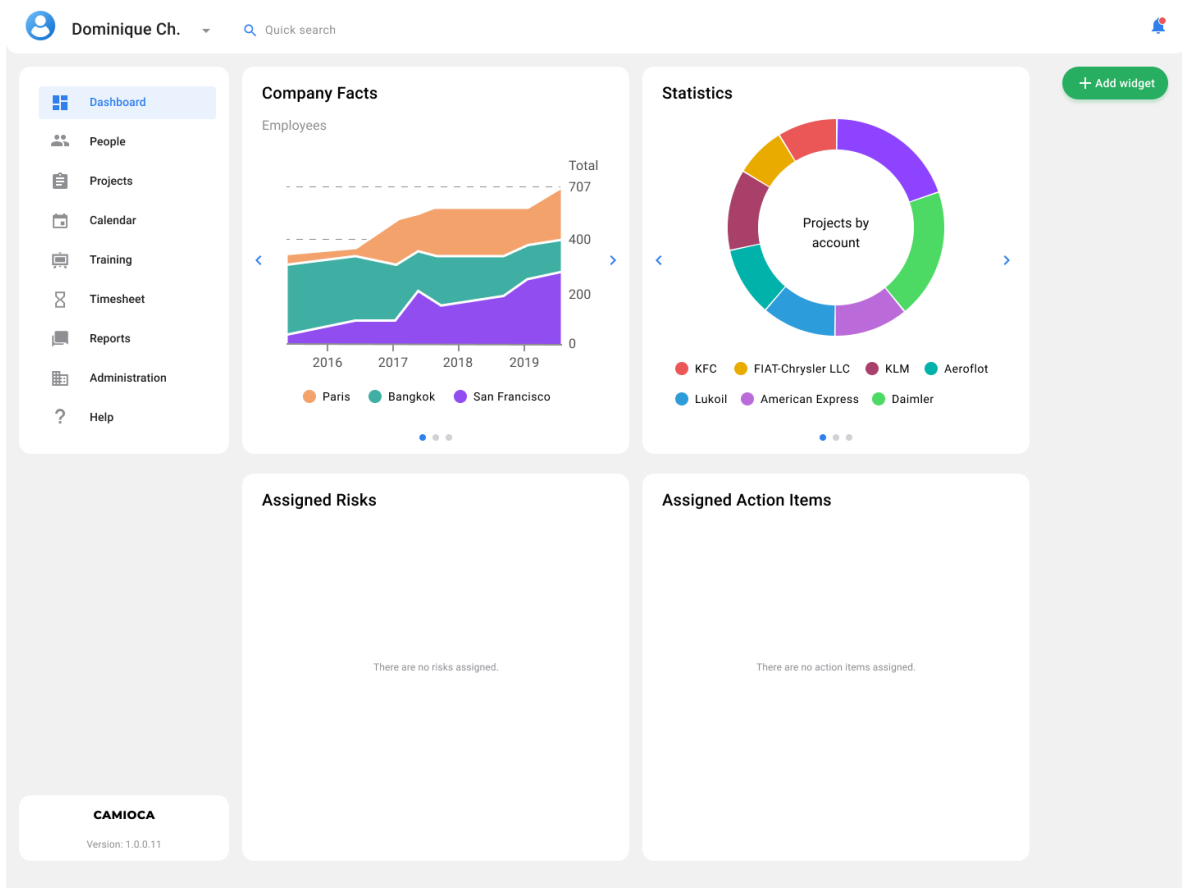


Figura 26 Dashboard

Dominique Ch.

Quick search

Dashboard

People

Projects

Calendar

Training

Timesheet

Reports

Administration

Help

Find project

All projects

Accounts

Filter

Project name

Search...

Budget type

Any

Short name

Search...

Status

In progress X

Project description

Search...

Notion: Full-text searching

Practice

Select

Account

Select

Start date from:

Select

Select

Source type

Select

MSA signer

Select

Finish date from:

Select

Select

Billing entity

Select

Industry

Select

Search mode

AND

OR

Search...

Region

Select

Clear

Search

CAMIOCA

Version: 1.0.0.11

Figura 27 Sitios

Dominique Ch.

Quick search

Dashboard

People

Projects

Calendar

Training

Timesheet

Reports

Administration

Help

Find person

All people

List of active people

Export to Excel

Delivery billable	Delivery unbillable	Administration	CEO	Sales	Marketing	Finance	HR	IT	Legal	PPS	L & D	OG	OBS	EBO
233	11	8	1	15	11	8	4	3	2	15	1	5	2	0

Delivery: 581 Non-delivery: 132 Total: 713

Filter

Regions

☒ UK (111)

☐ US (98)

☐ France (45)

Branches

☒ London (111)

☒ Liverpool (12)

☒ Manchester (5)

☒ Leeds (3)

☐ Cardiff (1)

Divisions

☒ Management (43)

☒ Finance (10)

☒ Legal (1)

☒ Operation (1)

☐ IT (1)

Clear

Found: 110

Full name	Branch	Division	User type	Contract type	Start date	ID
Konstantin Konstantinopolaky	London	Management	Delivery billable	Employee	12.01.2013	14321
Andrew Salgado	London	Management	Delivery billable	Employee	21.04.2014	21345
Magdalena Edinborough	Liverpool	Operation	Delivery billable	Subcontractor	02.11.2015	1894
Matt Travis	Liverpool	Management	Delivery billable	Employee	02.11.2016	135567
Daniel Wellington	Manchester	Operation	Delivery billable	Employee	02.11.2012	10973
Lucia Mirocini	Leeds	Finance	Delivery billable	Employee	02.11.2011	6324
Anette Brown	London	Operation	Delivery billable	Employee	02.11.2005	4322
Christian Lambrou	London	Finance	Delivery billable	Employee	02.11.2009	4521
Phil Collins	Manchester	Finance	Delivery billable	Subcontractor	02.11.2008	1422
Winston Holliday	London	Operation	Delivery billable	Employee	02.11.2017	45667
Carla Andrade	London	Operation	Delivery billable	Employee	02.11.2017	98871

Previous

12345...31Next

Show all

CAMIOCA

Version: 1.0.0.11

Figura 28 Modelos

44



Dominique Ch.

 Quick search



 Dashboard

 People

 Projects

 Calendar

 Training

 Timesheet

 Reports

 Administration

 Help

 Find project

 All projects

 Accounts

Project list

Industry

All Banking Financial Government Healthcare Hi-Tech Insurance ISV Logistics Media Pharma Public Retail Technology Telecom Utilities Internal Others

Status

☒ In progress ☒ Upcoming ☐ Suspended ☐ Completed ☐ Anticipated

In progress 581
Upcoming 2

Found: 5611



Practice	Project name	Short name	Status	Start date	End date	Account	Region	Industry	Source type	Billing entry
ENG	ABC app	ABCAPP	In progress	11.01.2008	11.01.2009	ABC	Latin America	Utilities	Customer	Corporation
ENG	ABC app	ABCAPP	In progress	11.01.2008	11.01.2009	ABC	Latin America	Utilities	Customer	Corporation
ENG	ABC app	ABCAPP	In progress	11.01.2008	11.01.2009	ABC	Latin America	Utilities	Customer	Corporation
ENG	ABC app	ABCAPP	In progress	11.01.2008	11.01.2009	ABC	Latin America	Utilities	Customer	Corporation
ENG	ABC app	ABCAPP	In progress	11.01.2008	11.01.2009	ABC	Latin America	Utilities	Customer	Corporation
ENG	ABC app	ABCAPP	In progress	11.01.2008	11.01.2009	ABC	Latin America	Utilities	Customer	Corporation
ENG	ABC app	ABCAPP	In progress	11.01.2008	11.01.2009	ABC	Latin America	Utilities	Customer	Corporation
ENG	ABC app	ABCAPP	In progress	11.01.2008	11.01.2009	ABC	Latin America	Utilities	Customer	Corporation
ENG	ABC app	ABCAPP	In progress	11.01.2008	11.01.2009	ABC	Latin America	Utilities	Customer	Corporation

← Previous

1 2 3 4 5 ... 31

Next →

Show all

CAMIOCA

Version: 1.0.0.11

Figura 29 Componentes

45

Dominique Ch.

Quick search

Dashboard

People

Projects

Calendar

Training

Timesheet

Reports

Administration

Help

Find person

All people

Filter

Name

Andrew Salgado

Email address

example@mail.com

Login

Login

Branch

Select

1C id

Id

Practice

Select

User type

Any

User contract type

Any

CV

Search...

☒ Capability
☒ Technology
☒ Certificate
☒ Course

Capabilities

Search...

☐ AND
☒ OR

Technology

Search...

☐ AND
☒ OR

Certificates

Search...

☐ AND
☒ OR

Courses

Search...

☐ AND
☒ OR

Education

Select

Search for specialization

☒ Show User Type, Login, Start Date, 1C Id
☐ Show Skype, Office phone, Mobile phone, Room number, Title

☐ Show current assignments
☐ Include terminated employees

Clear

Search

CAMIOCA

Version: 1.0.0.11

Figura 30 Configuraciones

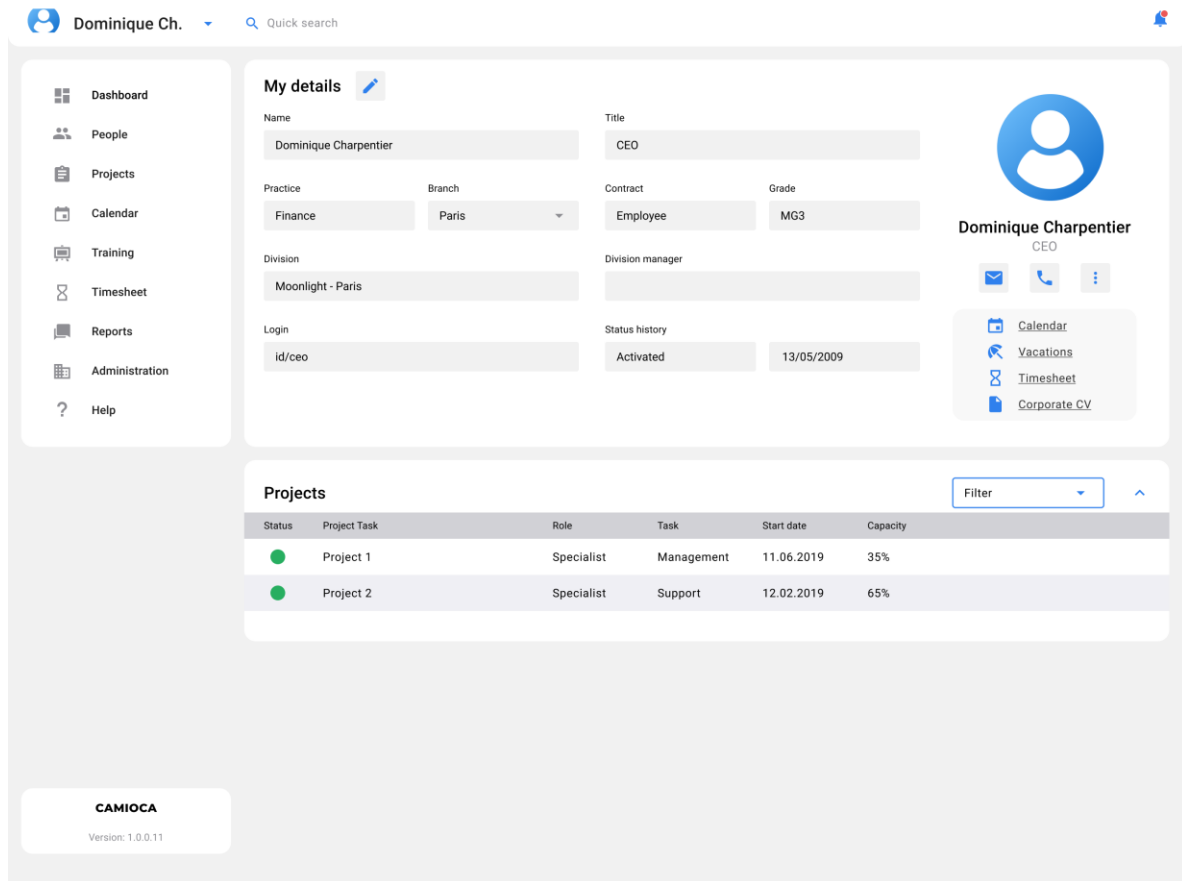


Figura 31 Configuración de perfil

Se aplicaron principios de accesibilidad, retroalimentación clara y consistencia con la identidad visual del sitio, manteniendo una disposición limpia y jerárquica que facilite el trabajo técnico.

3.7.4. Principios de UX/UI Aplicados

Durante el proceso de diseño se aplicaron los principios fundamentales de User Experience (UX) y User Interface (UI) para garantizar una experiencia coherente, predecible y accesible. El objetivo fue reducir la carga cognitiva del usuario y fomentar la comprensión intuitiva de cada acción dentro del sistema. A continuación, se describen los principales lineamientos implementados:

- **Consistencia visual y funcional**

Se mantuvo una coherencia estética en todos los elementos visuales: paleta de colores, tipografías, estilos de botones y espaciados. Esta consistencia refuerza la

identidad visual del sistema y facilita la familiarización del usuario con los patrones de interacción. Los componentes comunes (botones de acción, formularios, tablas) conservan el mismo comportamiento y estilo, asegurando uniformidad entre las diferentes secciones.

- **Jerarquía visual**

La información fue organizada de acuerdo con su nivel de importancia, empleando el contraste, tamaño y posición para guiar la atención del usuario. Los elementos primarios (botones principales o acciones críticas) se distinguen mediante colores llamativos, mientras que la información secundaria utiliza tonos neutros y tamaños tipográficos menores. Este enfoque contribuye a una navegación más fluida y a una mejor comprensión de las tareas disponibles.

- **Retroalimentación e interacción**

El sistema proporciona respuestas inmediatas a las acciones del usuario, ya sea mediante mensajes de éxito, alertas de error, cambios de color o animaciones sutiles.

Este principio de *feedback continuo* refuerza la sensación de control, evita la confusión y mejora la satisfacción general con el sistema.

- **Accesibilidad e inclusión**

Se aplicaron criterios de accesibilidad basados en las Pautas WCAG (**Web Content Accessibility Guidelines**), tales como contraste adecuado entre texto y fondo, tamaños de fuente legibles y compatibilidad con diferentes dispositivos. El diseño se adapta tanto a pantallas de escritorio como móviles (diseño responsivo), garantizando una experiencia homogénea. Además, se cuidó el uso de etiquetas claras y textos alternativos para elementos gráficos.

- **Usabilidad y minimalismo**

Cada pantalla fue diseñada con el principio de *menos es más*, eliminando elementos innecesarios que pudieran distraer al usuario. Se priorizó la simplicidad de las rutas

y la claridad de las acciones disponibles, reduciendo al mínimo el número de pasos requeridos para realizar una tarea.

3.7.5. Navegación entre Pantallas

El sistema fue diseñado bajo una arquitectura de navegación jerárquica y contextual, que permite al usuario desplazarse entre las distintas secciones sin perder orientación. Se implementó un menú principal accesible desde todas las vistas, complementado con rutas internas específicas según el rol del usuario.

El flujo general de navegación se establece de la siguiente manera:

1. **Visitante:** Accede a la página principal y puede explorar las secciones públicas como *Servicios*, *Galería* o *Contacto*. Desde esta vista puede iniciar sesión o registrarse para crear una cuenta.
2. **Cliente autenticado:** Una vez validado el acceso, el usuario puede visualizar su panel personal, revisar sus reservaciones o pedidos y actualizar su información de perfil. El menú lateral le permite regresar fácilmente al inicio o cerrar sesión.
3. **Administrador o encargado:** Accede al panel administrativo, donde dispone de un menú estructurado por módulos (Usuarios, Empresas, Servicios, Productos, Ventas y Configuración). Puede navegar entre secciones mediante pestañas o íconos de acceso rápido en el dashboard principal.

La estructura de navegación prioriza la simplicidad, reduciendo la cantidad de clics necesarios para alcanzar una acción concreta. Asimismo, se incorporaron rutas secundarias y breadcrumbs (*migas de pan*) que indican la posición actual dentro del sistema, evitando la desorientación.

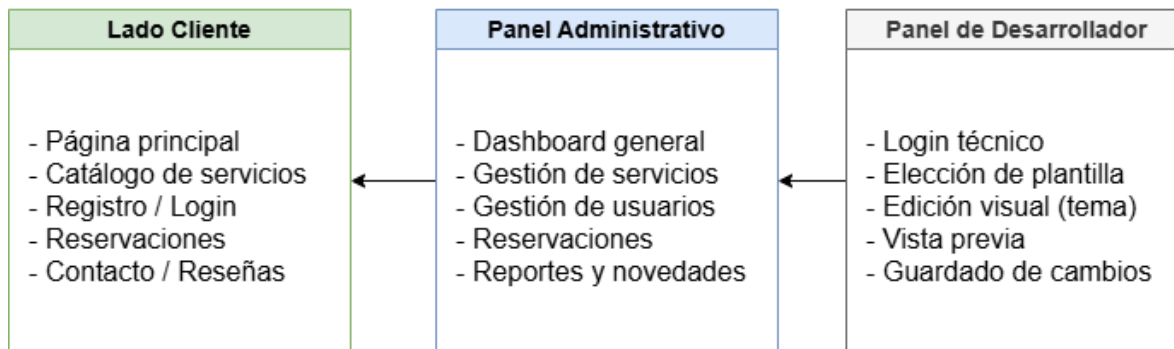


Figura 32 Navegación del sistema

3.7.6. Conclusión del diseño de interfaz

El uso de Figma y Balsamiq permitió establecer un proceso de diseño iterativo y colaborativo, en el que las pantallas fueron ajustadas con base en pruebas visuales y retroalimentación funcional. El resultado es una interfaz moderna, clara y consistente, que optimiza la experiencia de los distintos tipos de usuario. Este diseño, fundamentado en principios UX/UI, sienta las bases para la implementación visual final del sistema dentro del entorno web.

3.8. Conclusión del Diseño

El proceso de diseño del sistema representó una etapa fundamental dentro del ciclo de desarrollo, ya que permitió definir de forma estructurada la arquitectura, los componentes funcionales y la interacción visual de la aplicación antes de su implementación.

A través del análisis detallado de requerimientos, la elaboración de modelos conceptuales, la construcción del modelo entidad-relación y el diseño de las interfaces gráficas, se consolidó una base sólida para el desarrollo técnico posterior.

El diseño de la base de datos aseguró la organización lógica de la información mediante una estructura relacional normalizada, capaz de mantener la integridad, escalabilidad y coherencia de los datos entre los distintos módulos del sistema. Dicha estructura se construyó con un enfoque modular, permitiendo la gestión independiente de empresas, usuarios, servicios, productos y transacciones, al mismo tiempo que facilita la evolución futura del proyecto mediante la incorporación de nuevos componentes sin afectar la arquitectura existente.

Por otra parte, el diseño de la interfaz permitió visualizar anticipadamente la experiencia del usuario a través de prototipos funcionales que integran principios modernos de UX/UI. El uso de wireframes y mockups de alta fidelidad ayudó a validar la disposición de los elementos, la jerarquía visual, la navegabilidad y la consistencia estética del sistema antes de su desarrollo, reduciendo el riesgo de errores en la etapa de codificación.

Asimismo, la aplicación de principios de usabilidad, accesibilidad y retroalimentación garantizó que la interacción con el sistema sea clara, fluida y comprensible para los distintos perfiles de usuario. El enfoque centrado en el usuario (UCD) se reflejó en la construcción de flujos de navegación intuitivos, en la coherencia visual de los paneles y en la simplificación de las tareas más frecuentes.

En conjunto, las actividades de diseño técnico y visual sirvieron como puente entre la fase de análisis y la de desarrollo, transformando los requerimientos del sistema en representaciones tangibles y coherentes. El resultado de esta etapa es un modelo integral que combina estructura de datos, lógica de negocio y experiencia de usuario, garantizando que el sistema final sea eficiente, escalable y orientado a las necesidades reales de quienes lo utilizan.

4.2. Arquitectura del Sistema

La arquitectura del sistema fue diseñada bajo un enfoque modular y escalable, permitiendo la separación clara entre la interfaz de usuario, la lógica de negocio y el manejo de datos. El sistema se construyó siguiendo el paradigma Cliente–Servidor y el patrón Modelo–Vista–Plantilla (MVT) propio de Django, lo cual facilita su mantenimiento, reusabilidad y capacidad de adaptación a múltiples tipos de empresa dentro de un mismo entorno.

Esta arquitectura permite que el sistema funcione como una plataforma multi-empresa (multi-tenant), en la cual una empresa activa determina el contenido mostrado en el sitio público, los servicios ofertados y la configuración visual, mientras que el administrador de desarrollador cuenta con herramientas para

modificar, habilitar o deshabilitar módulos completos según las necesidades operativas del cliente.

4.2.1. Estructura General Cliente–Servidor

El sistema sigue el modelo clásico **Cliente–Servidor**, donde:

- El **cliente** corresponde al navegador web del usuario, encargado de mostrar la interfaz gráfica y ejecutar funciones de interacción mediante HTML, CSS y JavaScript.
- El **servidor** ejecuta la aplicación desarrollada en Django, procesa solicitudes HTTP, aplica la lógica de negocio y se comunica con la base de datos.
- La **base de datos** es responsable del almacenamiento estructurado de los datos, incluyendo usuarios, servicios, reservaciones, roles y configuraciones de empresa.

Flujo general de interacción:

1. El cliente realiza una solicitud al servidor mediante HTTP/HTTPS.
2. El servidor procesa la solicitud y consulta los datos si es necesario.
3. El servidor genera la respuesta aplicando plantillas y lógica.
4. El cliente presenta la información al usuario final.

4.2.2. Patrón Modelo–Vista–Plantilla (MVT) en Django

Django implementa el patrón **MVT**, el cual divide el sistema en tres componentes principales:

Tabla 16 Modelo-Vista-Plantilla

Componente	Descripción
Modelo (Model)	Representa las estructuras de datos. Define las entidades del sistema (usuarios, servicios, reservaciones, etc.) y gestiona su interacción con la base de datos.

Vista (View)	Contiene la lógica de negocio. Recibe solicitudes del cliente, realiza consultas, valida datos y selecciona las plantillas adecuadas para generar una respuesta.
Plantilla (Template)	Define la interfaz gráfica que se muestra al usuario. Se compone de archivos HTML, CSS y JavaScript, con etiquetas dinámicas propias de Django.

Este patrón permite que la lógica del sistema se mantenga separada de la presentación visual, facilitando la escalabilidad y mantenimiento del proyecto.

4.2.3. Diagrama de Arquitectura General

A continuación, se presenta el diagrama conceptual de la arquitectura del sistema:

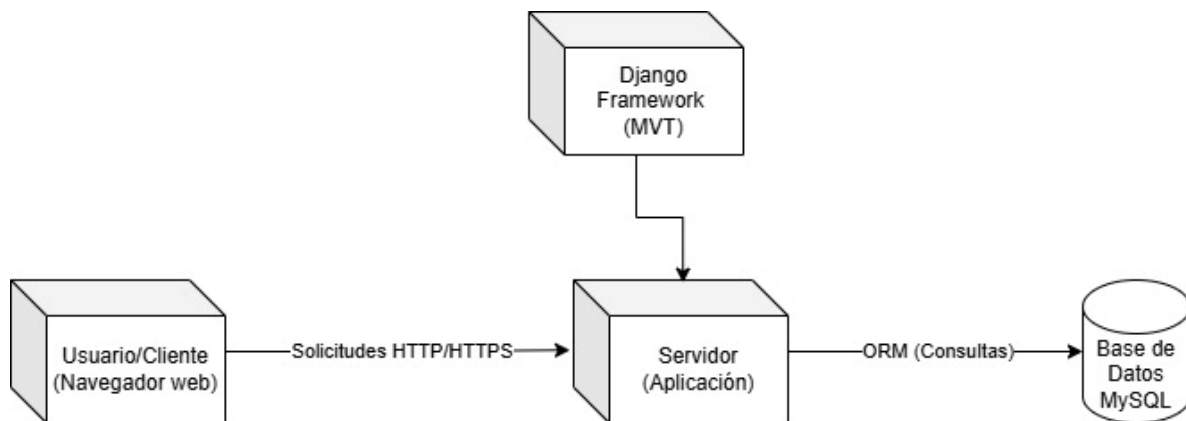


Figura 33 Arquitectura General

4.2.4. Herramientas y Frameworks Utilizados

Tabla 17 Herramientas y frameworks

Herramienta / Framework	Función dentro del sistema
Django 5	Framework principal para el desarrollo del sistema web, utilizando el patrón Modelo–Vista–Plantilla (MVT), el ORM

	integrado y el manejo de autenticación y permisos.
Bootstrap 5	Framework de diseño para la creación de una interfaz gráfica responsiva y adaptable a múltiples dispositivos.
JavaScript	Permite añadir interactividad al cliente, manejo dinámico de formularios, alertas, modales y validaciones.
MySQL	Sistema de gestión de base de datos utilizado para almacenar la información estructurada del sistema, como usuarios, servicios, reservaciones y configuraciones.
PayPal SDK	Implementado para la integración con la pasarela de pagos, permitiendo procesar transacciones en línea de forma segura.
ReportLab / xhtml2pdf	Utilizadas para la generación automática de reportes en formato PDF dentro del sistema.
Git + Visual Studio Code / Cursor	Herramientas para el control de versiones y entorno de desarrollo.

4.3. Estructura del Proyecto

La estructura interna del proyecto fue organizada siguiendo los lineamientos y buenas prácticas recomendadas para el desarrollo de aplicaciones en Django. La organización modular permite dividir el sistema en componentes independientes, facilitando su mantenimiento, escalabilidad y la incorporación futura de nuevas funcionalidades. Además, esta estructura permite que la lógica de negocio, la gestión de datos y la interfaz de usuario se mantengan claramente separadas, favoreciendo un desarrollo limpio y ordenado.

El proyecto completo se encuentra contenido dentro de un directorio principal llamado **xoleongo**, el cual actúa como raíz del sistema. Dentro de este directorio se encuentran tanto los archivos de configuración global como las diferentes aplicaciones (apps) que componen el sistema.

4.3.1. Organización General de Carpetas

A continuación, se presenta la estructura general del proyecto:

xoletongo/

```
|
|
| — manage.py          # Punto de inicio del proyecto Django
| — requirements.txt    # Librerías utilizadas en el sistema
| — README.md          # Información técnica del proyecto
|
| — xoletongo/          # Configuración global del proyecto
|   | — settings.py     # Configuración de apps, BD, plantillas, seguridad
|   | — urls.py         # Enrutamiento principal
|   | — wsgi.py         # Configuración para despliegue en servidor web
|   | — asgi.py         # Configuración para entornos asincrónicos
|
| — adminpanel/         # Panel administrativo de la empresa (operaciones
internas)
|   | — models.py
|   | — views.py
|   | — forms.py
|   | — urls.py
|   | — utils.py
|   | — context_processors.py
|   | — middleware.py
```

```

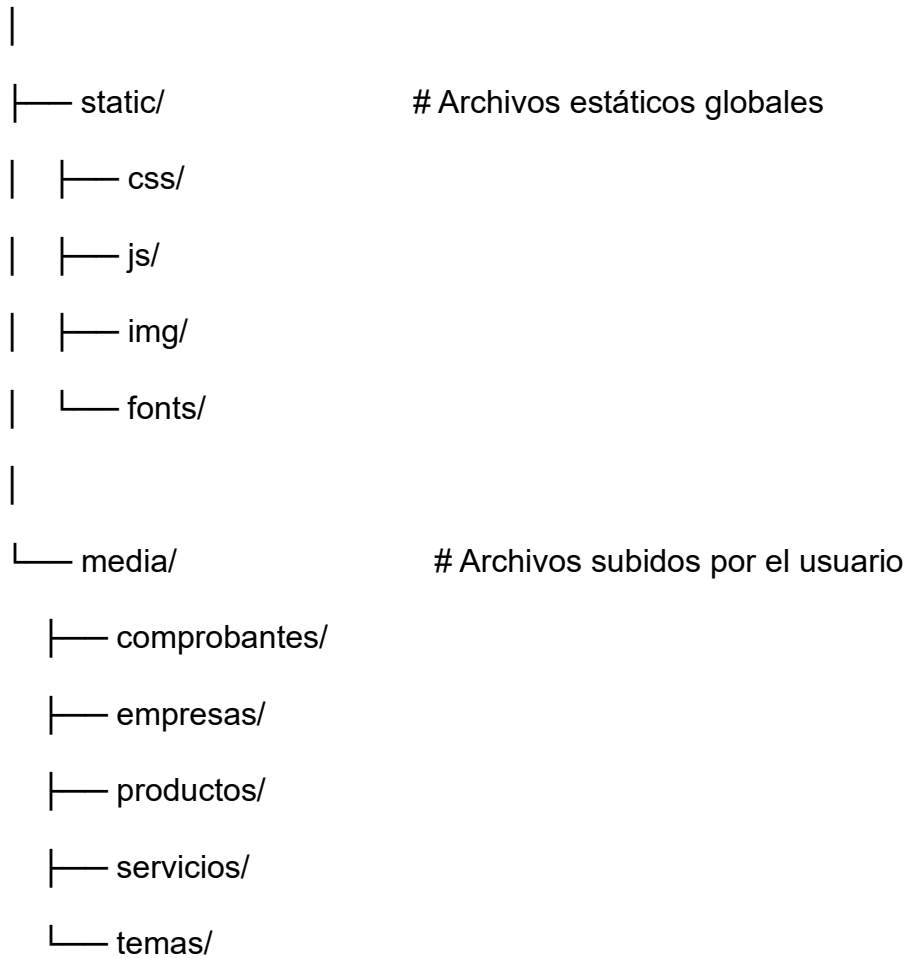
|   ├── templates/
|   └── static/
|
|   ├── devpanel/           # Panel administrativo del desarrollador
|   │   ├── models.py
|   │   ├── views.py
|   │   ├── urls.py
|   │   ├── context_processors.py
|   │   ├── management/
|   │   ├── templates/
|   │   └── static/
|   │
|   ├── empresas/          # Administración de empresas y multi-tenant
|   │   ├── models.py      # Empresa, temas, módulos activos
|   │   ├── backends.py
|   │   ├── signals.py
|   │   ├── management/
|   │   └── utils.py
|   │
|   ├── home/              # Sitio público y páginas visibles al usuario final
|   │   ├── models.py
|   │   ├── views.py
|   │   └── urls.py

```

```

|   └─ templates/
|
|   └─ usuarios/                # Gestión de autenticación y roles
|       └─ models.py
|       └─ forms.py
|       └─ urls.py
|       └─ admin.py
|
|   └─ servicios/              # Módulo de servicios de la empresa
|       └─ models.py
|       └─ views.py
|       └─ urls.py
|       └─ templates/
|
|   └─ reservaciones/          # Flujo de reservaciones y control de disponibilidad
|       └─ models.py
|       └─ views.py
|       └─ urls.py
|       └─ templates/
|
|   └─ templates/              # Plantillas base globales
|       └─ base.html
|       └─ base_template.html

```



Esta organización presenta una separación clara de propósitos:

- **Configuración global** (xoleongo/)
- **Apps funcionales** (como adminpanel/, devpanel/, reservaciones/, etc.)
- **Plantillas base** (templates/)
- **Archivos estáticos y multimedia** (static/ y media/)

De esta forma, el sistema sostiene un orden lógico que beneficia el desarrollo y mantenimiento continuo.

4.3.2. Aplicaciones Principales del Sistema

El proyecto se compone de diversas aplicaciones internas (apps), cada una diseñada para manejar una parte específica del funcionamiento del sistema. Las más relevantes son:

4.3.2.1. *home* → *Sitio público y reservaciones*

Esta aplicación contiene la lógica y las vistas dirigidas al usuario final, incluyendo:

- Página de inicio
- Presentación de servicios
- Sección de reservaciones
- Sección de contacto
- Visualización de información de la empresa activa

Su objetivo principal es ofrecer una interfaz sencilla y clara que permita a los visitantes navegar por los servicios disponibles y realizar reservaciones conforme a la disponibilidad del negocio.

4.3.2.2. *usuarios* → *Autenticación y roles*

En esta aplicación se administra:

- Registro e inicio de sesión
- Recuperación de contraseña
- Perfiles de usuario
- Roles y permisos
- Asociación del usuario con una empresa en casos específicos

El modelo de usuario fue extendido para incluir atributos adicionales necesarios para el funcionamiento del sistema, además de implementar roles diferenciados según el tipo de panel al que el usuario puede acceder.

4.3.2.3. *adminpanel* → *Panel administrativo interno*

Este módulo corresponde al entorno de administración disponible únicamente para los usuarios pertenecientes a la empresa activa. Dentro del panel es posible realizar acciones como:

- Gestión de servicios y productos
- Control de reservaciones
- Gestión de clientes y empleados
- Modificación del estado de ventas y comprobantes
- Generación de reportes
- Visualización de métricas internas

Su propósito es permitir la operación diaria del negocio desde una interfaz organizada y con herramientas diseñadas para facilitar los procesos internos de la empresa.

4.3.2.4. *developerpanel* → *Panel de desarrollador*

Esta aplicación está dirigida exclusivamente al desarrollador encargado del sistema. Las herramientas disponibles en este módulo incluyen:

- Gestión de empresas registradas
- Activación de una empresa para uso del sistema
- Configuración visual (logos, colores, tipografías)
- Activación o desactivación de módulos (servicios, productos, reseñas)
- Administración de roles personalizados
- Control sobre archivos estáticos y contenido multimedia
- Herramientas de mantenimiento y respaldo de datos

Con este módulo, el sistema se convierte en una plataforma robusta, capaz de adaptarse a distintos giros y permitir la administración completa desde un punto centralizado.

4.3.3. Configuración de Rutas y Vistas Principales

El archivo `urls.py` ubicado en la carpeta principal `xoleongo/` es el encargado de administrar las rutas globales del proyecto. Su función es redirigir cada solicitud hacia la aplicación correspondiente. El archivo contiene una estructura similar a:

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path("", include('home.urls')),  
    path("", include('servicios.urls')),  
    path('usuarios/', include('usuarios.urls')),  
    path("", include('reservaciones.urls')),  
    path('adminpanel/', include('adminpanel.urls')),  
    path('dev/', include('devpanel.urls')),  
]
```

Gracias a este diseño, cada módulo puede manejar sus rutas de forma independiente, manteniendo el sistema ordenado y escalable.

Las vistas dentro de cada aplicación gestionan:

- Solicitudes GET y POST
- Validación de datos
- Renderización de plantillas
- Redirecciones
- Control de acceso mediante decoradores o middleware

Esto permite un control preciso sobre la interacción entre el usuario y el sistema.

4.3.4. Integración de Plantillas Base y Herencia de Vistas

El proyecto utiliza dos plantillas base ubicadas en la carpeta global `templates/`:

- **base.html** → utilizada para el sitio público
- **base_template.html** → utilizada en los paneles administrativos

Ambas plantillas implementan bloques (`{% block content %}`) que permiten que las vistas específicas hereden la estructura visual principal, los estilos generales y los elementos comunes como:

- Barra de navegación
- Cabeceras
- Mensajes de notificación
- Scripts y hojas de estilo globales
- Pie de página

El mecanismo de herencia se aplica de la siguiente forma:

```
{% extends "base.html" %}
```

```
{% block content %}
```

```
<div class="container">
```

```
    <h1>Bienvenido al sitio</h1>
```

```
</div>
```

```
{% endblock %}
```

Este patrón fomenta la reutilización de código, evita duplicidades y permite que cualquier modificación al estilo general del sistema se refleje automáticamente en todas las vistas heredadas.

4.5. Desarrollo del Panel Administrativo Interno

El Panel Administrativo Interno es uno de los componentes centrales del sistema, ya que proporciona a la empresa activa las herramientas necesarias para gestionar su operación diaria. Este panel fue diseñado para ser utilizado únicamente por el

personal interno de la organización, quienes cuentan con los permisos necesarios para acceder a las funciones correspondientes a su rol.

A diferencia del sitio público, donde el usuario final únicamente puede visualizar servicios y realizar reservaciones, el panel administrativo permite controlar de manera integral los elementos clave del negocio: clientes, servicios, productos, ventas, reservaciones, comprobantes de transferencia, estado de pagos, reportes y la configuración operativa general.

Su diseño se basa en una interfaz clara, segmentada por módulos y con acceso restringido mediante un sistema de roles. Este panel está implementado en la aplicación `adminpanel/` y su función es facilitar al personal de la empresa un conjunto de herramientas intuitivas y eficientes para la gestión interna.

4.5.1. Descripción General del Módulo

El panel administrativo ofrece una interfaz operativa donde los usuarios con permisos (Administrador, Empleado, Encargado) pueden gestionar los diferentes recursos del sistema.

Su construcción se basó en tres principios fundamentales:

1. **Operatividad:** permitir a la empresa realizar tareas administrativas sin complicaciones.
2. **Seguridad:** restringir el acceso según el rol del usuario y la empresa activa.
3. **Escalabilidad:** posibilitar la incorporación de nuevos módulos sin reestructurar el panel.

El diseño del panel incluye:

- Barra lateral (sidebar) con módulos organizados.
- Dashboard general con indicadores clave.
- Tablas de gestión con funciones de búsqueda y filtrado.
- Formularios dinámicos ligados a los modelos de Django.

- Vistas para revisión de pagos por transferencia.
- Secciones para modificar estados de reservación.

Cada vista está asociada a una función específica y presenta únicamente la información necesaria según el rol del usuario.

4.5.2. Control de Acceso y Roles de Usuario

El control de acceso es una pieza fundamental del panel administrativo, ya que impide que un usuario sin permisos pueda ver o gestionar información confidencial.

La aplicación de usuarios implementa un modelo extendido que incluye roles como:

- **Administrador**
- **Empleado**
- **Encargado**

Cada uno posee permisos definidos, los cuales determinan qué secciones del panel puede visualizar y modificar.

Mecanismos utilizados para la protección del panel

1. **Decoradores** **(@login_required)**
Garantizan que solo usuarios autenticados puedan acceder al panel.
2. **Middlewares** **personalizados**
Validan que el usuario pertenezca a la empresa activa.
3. **Verificación** **de** **rol**
Solo usuarios con rol interno pueden acceder al módulo adminpanel.
4. **Restricciones** **en** **vistas**
Se emplean condiciones para mostrar u ocultar opciones dentro del panel según el tipo de usuario.

Ejemplo de lógica aplicada:

- Un *Empleado* puede registrar reservaciones y subir comprobantes, pero no puede editar la información de la empresa.
- Un *Encargado* puede modificar estados de reservaciones.
- Un *Administrador* tiene acceso a todas las funciones del panel, incluyendo reportes y métricas.

Esto permite un uso ordenado y seguro del sistema.

4.5.3. Gestión de Entidades Principales

El panel administrativo está dividido en varias secciones que permiten gestionar las entidades principales del negocio. Entre las más importantes se encuentran:

Gestión de clientes

El personal administrativo puede:

- Consultar la lista completa de clientes registrados.
- Filtrar por nombre, correo o teléfono.
- Acceder al historial de reservaciones de cada cliente.
- Modificar o corregir información básica en caso de ser necesario.

Esta sección permite mantener un registro organizado y detallado de todos los usuarios que han interactuado con la empresa.

Gestión de empleados

El panel permite:

- Registrar empleados nuevos.
- Asignar roles internos.
- Activar o desactivar cuentas.
- Editar información de acceso.

Con ello se crea una jerarquía clara dentro del personal operativo.

Gestión de servicios y productos

Dentro del panel es posible:

- Agregar nuevos servicios o productos ofrecidos por la empresa.
- Modificar precios, duraciones, modalidades y disponibilidad.
- Administrar imágenes representativas.
- Activar o desactivar servicios sin eliminarlos del sistema.

Esta sección es fundamental para mantener actualizado el catálogo empresarial.

Gestión interna de reservaciones

Esta funcionalidad permite administrar el ciclo de vida de las reservaciones recibidas, tanto las pagadas por PayPal como aquellas pagadas por transferencia bancaria.

El sistema permite:

- Consultar todas las reservaciones realizadas.
- Ver detalles como fecha, cliente, servicio, monto pagado y estado.
- Cambiar el estado de la reservación según el flujo operativo:
 - Pendiente
 - Pagado
 - En revisión
 - Confirmado
 - Rechazado
 - Cancelado
 - Completado

Esta gestión garantiza que el personal de la empresa pueda llevar un control adecuado de todas las operaciones.

Validación de pagos por transferencia

Este módulo es exclusivo del panel administrativo y se desarrolló específicamente para validar comprobantes de pago enviados por los usuarios.

Incluye:

- Vista detallada de cada reservación pendiente de transferencia.
- Visualización del comprobante (imagen o PDF).
- Descarga del archivo para validación externa.
- Botones para aprobar o rechazar el pago.
- Registro automático de la decisión tomada.

Este proceso permite a la empresa validar pagos de forma segura y organizada.

Gestión de ventas

Dependiendo del giro de la empresa, el panel puede mostrar:

- Total de reservas confirmadas
- Total de ingresos recibidos
- Reservas completadas
- Reservas pendientes

Esta información se utiliza posteriormente en los reportes y métricas.

4.5.4. Generación de Reportes en PDF

El panel integra herramientas para generar reportes profesionales en formato PDF utilizando librerías como ReportLab o xhtml2pdf.

Estos reportes pueden incluir:

- Ventas por periodo
- Reservas confirmadas
- Listado de clientes

- Movimientos del sistema
- Servicios utilizados con mayor frecuencia

La generación automática de reportes permite a la empresa documentar su operación y analizar su actividad sin realizar cálculos manuales.

Características de los reportes

- Diseño uniforme y encabezado con el logo de la empresa activa.
- Tablas con datos organizados.
- Fechas exactas de generación.
- Formato A4 para impresión o envío por correo.

4.5.5. Panel de Métricas e Indicadores

Con el objetivo de proporcionar una visión global del rendimiento de la empresa, el panel incluye un módulo de métricas que muestra:

- Total de reservaciones.
- Reservaciones completadas.
- Reservaciones pendientes o en revisión.
- Servicios más reservados.
- Ingresos generados.
- Clientes nuevos registrados en un periodo.

Estos datos se presentan mediante:

- Gráficas de barras.
- Indicadores numéricos.
- Tarjetas de estado (cards).

El panel se convierte así en una herramienta estratégica para la toma de decisiones.

4.5.6. Registro de Novedades o Auditorías

Para reforzar la transparencia y control interno, el sistema implementa un registro de acciones importantes, el cual funciona como una bitácora de eventos.

Este registro incluye:

- Cambios en el estado de reservaciones.
- Verificación o rechazo de comprobantes.
- Modificaciones de servicios.
- Actualización de información de clientes o empleados.
- Generación de reportes.
- Inicios de sesión y cierres de sesión (si se habilita).

La auditoría permite tener rastreo completo de los movimientos internos del sistema.

4.6. Desarrollo del Panel de Desarrollador

El Panel de Desarrollador constituye el nivel más alto de administración dentro del sistema. Este módulo fue diseñado para que únicamente el usuario con rol de desarrollador o administrador maestro pueda acceder a configuraciones avanzadas que afectan el comportamiento global de la plataforma.

A diferencia del panel administrativo interno (orientado a la operación diaria de una empresa), el panel de desarrollador gestiona aspectos estructurales del sistema, tales como:

- Registro y administración de múltiples empresas.
- Activación de la empresa que estará operando públicamente.
- Configuración visual completa del sitio (tema, colores, logos, tipografías).
- Administración y creación de roles personalizados.
- Control de módulos habilitados (servicios, tienda, reseñas).
- Cuidado del contenido multimedia y respaldo de datos.

Este módulo convierte al sistema en una plataforma multiempresa y multimódulo, capaz de adaptarse a diversos tipos de negocios sin necesidad de modificar el código fuente.

4.6.1. Descripción General y Propósito

El panel de desarrollador tiene como objetivo principal permitir la configuración centralizada del sistema, otorgando herramientas para definir la estructura empresarial y la apariencia visual del sitio público.

Su propósito es garantizar que:

1. La plataforma pueda ser utilizada por diferentes empresas sin modificar la programación.
2. El desarrollador tenga control total de módulos, roles y configuraciones.
3. La empresa activa pueda cambiarse sin comprometer datos ni estabilidad.
4. El sistema se mantenga escalable y adaptable a nuevos requerimientos.

La aplicación devpanel/ contiene todos los modelos, vistas, formularios y utilidades relacionadas con esta administración avanzada.

Entre sus funciones más relevantes se encuentran:

- Registrar empresas nuevas.
- Activar una empresa como “empresa principal”.
- Configurar datos institucionales.
- Personalizar tema visual (identidad gráfica).
- Definir roles personalizados y permisos.
- Activar o desactivar módulos globales del sistema.
- Gestionar archivos estáticos y contenido multimedia.
- Descargar respaldos, bases de datos o archivos en formato CSV.

El panel está protegido mediante control estricto de permisos, permitiendo únicamente al usuario desarrollador acceder a sus funciones.

4.6.2. Configuración Institucional del Sistema

Una de las características más importantes del panel de desarrollador es la posibilidad de configurar la empresa activa, es decir, aquella empresa cuyos servicios, logo, tema y contenido se mostrarán en el sitio público.

Administración de empresas

El panel permite:

- Registrar una nueva empresa con información básica:
 - Nombre comercial
 - Nombre del titular
 - Descripción
 - Teléfono
 - Dirección
 - Correo institucional
- Editar empresas existentes.
- Desactivar o eliminar empresas que ya no se utilizarán.

Activación de la empresa

El desarrollador puede seleccionar cuál empresa estará activa dentro del sistema. Esta acción:

- Desactiva automáticamente a cualquier otra empresa activa.
- Cambia el contenido del sitio público a la información de la empresa seleccionada.
- Aplica el tema visual correspondiente.
- Muestra únicamente los servicios y configuraciones pertenecientes a esa empresa.

Configuración visual del sitio

El panel incluye una sección destinada a la personalización completa de la identidad gráfica:

- Logo principal
- Logo en modo oscuro
- Paleta de colores (primario, secundario, acentos)
- Tipografías
- Imágenes promocionales para el hero
- Iconografía institucional
- Banners y galerías
- Colores secundarios para navbar, pie de página, botones, etc.

Estas configuraciones se almacenan en la base de datos y se aplican dinámicamente mediante context processors.

4.6.3. Gestión Avanzada de Usuarios y Permisos

El panel de desarrollador permite gestionar los roles que serán utilizados tanto por el panel interno como por el sistema público.

Creación de roles personalizados

El desarrollador puede crear y configurar roles con permisos específicos. Algunos roles predeterminados son:

- DeveloperAdmin
- CompanyAdmin
- Empleado
- Encargado
- Cliente

Sin embargo, el panel permite añadir roles personalizados adaptados a cada empresa.

Asignación de permisos

Cada rol puede incluir permisos como:

- Permiso para gestionar reservaciones
- Permiso para validar pagos por transferencia
- Permiso para gestionar empleados
- Permiso para modificar contenido institucional
- Permiso para acceder al panel administrativo

El sistema utiliza el modelo extendido de usuarios para asignar roles que luego son verificados mediante decoradores, middlewares o funciones personalizadas.

Activación y desactivación de cuentas

El desarrollador puede:

- Activar o desactivar usuarios del sistema.
- Cambiar roles de manera manual.
- Transferir permisos especiales entre usuarios.

Esta sección garantiza el control total sobre la estructura operativa del sistema.

4.6.4. Módulo de Servicios Dinámicos

Este módulo fue diseñado para que la empresa pueda manejar sus servicios sin depender del desarrollador.

El sistema permite cargar servicios a partir de:

- Formularios internos del panel.
- Archivos en formato JSON (si se desea).
- Configuraciones dinámicas por parte de la empresa.

El panel de desarrollador puede:

- Definir el esquema o estructura de los servicios.
- Añadir campos personalizados (descripción extendida, políticas, imágenes adicionales).
- Establecer categorías o tipos de servicio.
- Habilitar o deshabilitar temporalmente servicios globales.

Esta versatilidad es particularmente útil en giros como turismo, hotelería o comercio electrónico.

4.6.5. Administración de Medios Estáticos

El panel permite gestionar los archivos que la empresa utiliza en su sitio público:

- Logos
- Fotografías de productos o servicios
- Iconos
- Banners
- Archivos de ayuda
- PDFs o imágenes institucionales

El desarrollador puede revisar, eliminar o sustituir recursos de manera centralizada.

Esto es importante porque:

- Evita acumulación de archivos obsoletos.
- Optimiza el rendimiento del servidor.
- Mantiene consistencia visual entre empresas.
- Reduce errores al cambiar el contenido institucional.

Todos estos archivos se almacenan en la carpeta media/ organizada por categorías para facilitar su administración.

4.6.6. Copias de Seguridad y Mantenimiento

El panel de desarrollador incluye herramientas esenciales para la integridad de la información:

Exportación de base de datos o datos relevantes

El sistema permite:

- Exportar datos de clientes
- Exportar reservaciones
- Exportar servicios
- Descargar la base de datos o una copia parcial

Esto es útil para auditorías, migraciones o respaldos externos.

Respaldo de archivos multimedia

El desarrollador puede generar un respaldo de los contenidos almacenados en:

- media/empresas/
- media/servicios/
- media/productos/
- media/comprobantes/

Estos respaldos pueden almacenarse de manera local o en un almacenamiento externo.

Mantenimiento general

El panel incluye acciones como:

- Limpieza de archivos obsoletos
- Regeneración de configuraciones
- Actualización de logotipos o elementos globales
- Sincronización del tema con la empresa activa

Con estas herramientas, el sistema puede mantenerse estable durante largos periodos de tiempo y prevenir errores derivados de acumulación de datos.

4.7. Pruebas y Validación del Sistema

La fase de pruebas es una etapa fundamental en el desarrollo del sistema, ya que permite verificar el correcto funcionamiento de cada módulo, identificar errores y asegurar que la plataforma cumpla con los requisitos funcionales, operativos y de seguridad establecidos durante la etapa de análisis.

En este apartado se describen los tipos de pruebas aplicadas, los escenarios evaluados y la retroalimentación obtenida durante la validación del sistema con usuarios reales y con el personal administrativo de la empresa activa.

El objetivo de esta fase es garantizar que el sistema sea robusto, eficiente, fácil de usar y capaz de operar en un entorno real sin comprometer la experiencia del usuario ni la integridad de los datos.

4.7.1. Tipos de Pruebas Aplicadas

Durante el desarrollo del sistema se aplicaron diferentes tipos de pruebas para abarcar tanto aspectos técnicos como operativos y de usabilidad. Entre las más relevantes se encuentran:

4.7.1.1. Pruebas Unitarias

Las pruebas unitarias consistieron en evaluar funciones o componentes individuales del sistema, especialmente aquellos relacionados con:

- Lógica de reservaciones
- Validación de disponibilidad de servicios
- Cálculo de fechas y horarios
- Procesamiento de pagos (PayPal / transferencia)
- Validación de formularios

- Control de permisos por rol

Estas pruebas permitieron verificar que cada fragmento de código ejecutara correctamente su lógica sin depender de otros módulos.

4.7.1.2. Pruebas Funcionales

Las pruebas funcionales se enfocaron en validar que cada módulo cumpliera con su propósito dentro del sistema. Esto incluyó:

- Flujo completo de registrarse e iniciar sesión
- Proceso de reservación (selección → fecha → confirmación → pago)
- Subida de comprobantes de transferencia
- Cambio de estado de reservaciones desde el panel administrativo
- Creación, edición y eliminación de servicios
- Configuración institucional desde el panel del desarrollador
- Activación de empresa y aplicación automática del tema visual

Estas pruebas fueron esenciales para garantizar que el sistema respondiera de manera adecuada a las interacciones del usuario final.

4.7.1.3. Pruebas de Seguridad

Debido a que el sistema almacena datos sensibles (usuarios, pagos y comprobantes), se realizaron pruebas orientadas a validar aspectos de seguridad como:

- **Protección de rutas mediante roles** (login requerido, comprobación de permisos).
- **Redirección automática** cuando un usuario intenta acceder a una sección restringida.
- **Validación de formatos de archivos** en la subida de comprobantes.
- **Hashing de contraseñas** mediante el sistema nativo de Django.
- **Uso de tokens seguros** en recuperación de contraseña.

- **Restricción de acceso** a paneles administrativos según la empresa activa.

Estas pruebas garantizan que solo los usuarios autorizados puedan realizar acciones sensibles.

4.7.1.4. Pruebas de Usabilidad

Con el objetivo de mejorar la experiencia del usuario se realizaron pruebas con personas ajenas al desarrollo, quienes evaluaron:

- Claridad de textos y botones
- Flujo del proceso de reservación
- Facilidad para encontrar información
- Comprensibilidad del método de pago
- Accesibilidad desde dispositivos móviles
- Navegación entre panel del cliente y sitio público

Los resultados de estas pruebas permitieron mejorar la interfaz, ajustar espaciados y reorganizar elementos de navegación para lograr una interacción más fluida.

4.7.2. Escenarios de Prueba

Durante la validación se diseñaron y ejecutaron diferentes escenarios que simulan situaciones reales que pueden ocurrir durante el uso del sistema. A continuación, se describen algunos de los más relevantes.

Escenario 1: Registro e inicio de sesión

Objetivo: verificar el correcto funcionamiento del sistema de autenticación.

Resultados esperados: creación de cuenta, validación de datos, inicio de sesión y acceso al panel personal.

Resultado obtenido: el sistema cumplió correctamente todas las etapas; se mejoró el mensaje en caso de error de contraseña.

Escenario 2: Realizar una reservación con pago por PayPal

Objetivo: asegurar que el usuario pueda completar todo el flujo sin errores.

Pasos evaluados:

1. Selección del servicio
2. Selección de fecha
3. Confirmación
4. Pago
5. Redirección al sistema
6. Registro automático de reservación

Resultado obtenido: el proceso fue exitoso; la reservación se registró adecuadamente con su ID de transacción.

Escenario 3: Pago por transferencia bancaria

Objetivo: validar la subida de comprobante y registro del estado inicial.

Evaluaciones realizadas:

- Visualización de datos bancarios de la empresa
- Subida de imagen o PDF
- Registro de reservación en estado *Pendiente de validación*

Resultado

obtenido:

El proceso se ejecutó correctamente. Se fortalecieron las restricciones de formato y tamaño del archivo.

Escenario 4: Validación de comprobante desde el panel administrativo

Objetivo: permitir al personal administrativo aprobar o rechazar un pago por transferencia.

Pasos evaluados:

- Visualización del comprobante

- Revisión manual
- Cambio de estado de la reservación
- Notificación correspondiente (si aplica)

Resultado:

Funcionamiento exitoso. Se ajustó el diseño de la vista para facilitar la lectura de comprobantes.

Escenario 5: Gestión de servicios desde el panel administrativo

Objetivo: verificar la creación, edición y eliminación.

Resultado:

Correcto funcionamiento. Se mejoró la validación de campos obligatorios.

Escenario 6: Cambio de empresa activa desde el panel del desarrollador

Objetivo: asegurar que el sistema actualice tema, contenido y servicios al cambiar la empresa activa.

Resultado:

El cambio se aplicó correctamente. El tema visual y los servicios se mostraron conforme a la empresa seleccionada.

4.7.3. Retroalimentación del Usuario o Empresa

Durante la fase de pruebas, se obtuvo retroalimentación directa del personal encargado de operar el sistema y de usuarios finales que realizaron reservaciones simuladas. La retroalimentación permitió identificar oportunidades de mejora y confirmar las fortalezas del sistema.

Algunos aspectos destacados fueron:

Opiniones positivas

- La interfaz del sitio público fue descrita como clara, fácil de usar y visualmente agradable.

- El flujo de reservaciones resultó intuitivo, especialmente con PayPal.
- La opción de pago por transferencia brindó confianza a usuarios sin tarjeta.
- El panel administrativo permitió gestionar reservaciones de forma práctica.
- La subida y validación de comprobantes facilitó el control financiero.
- El tema visual personalizable permitió adecuar la identidad de la empresa.

Oportunidades de mejora sugeridas

- Implementar notificaciones automáticas por correo al aprobar comprobantes.
- Añadir filtros avanzados para buscar reservaciones por rango de fechas.
- Mejorar la vista móvil del panel administrativo.
- Integrar métricas adicionales para facilitar decisiones operativas.

La retroalimentación recibida permitió realizar ajustes oportunos, lo que garantizó un producto final más estable, usable y adecuado a las necesidades de la empresa.

4.8. Despliegue del Sistema

El despliegue del sistema corresponde a la etapa en la cual la aplicación se prepara para operar en un ambiente real de producción, accesible al público general a través de un dominio y un servidor. Aunque el sistema se encuentra funcional en su entorno local, al momento de elaboración de este documento aún no ha sido desplegado en un servidor de producción, ya que el proceso depende de la aprobación del cliente, la asignación de dominio y la definición del servicio de infraestructura que será utilizado.

No obstante, para garantizar la futura puesta en producción del sistema, se documenta a continuación el proceso de preparación del entorno, la configuración necesaria y las recomendaciones de servidores donde puede ser alojado.

4.8.1. Preparación del Entorno de Producción

Antes de realizar el despliegue, fue necesario preparar el proyecto para operar bajo un entorno óptimo de producción. Esta preparación incluyó ajustes a nivel de configuración, seguridad, base de datos y estructura del servidor.

Configuración de Variables de Entorno

El archivo settings.py fue configurado para utilizar variables de entorno protegidas, tales como:

- SECRET_KEY
- Credenciales de base de datos (usuario, contraseña, host)
- Credenciales de la API de PayPal (PAYPAL_CLIENT_ID, PAYPAL_CLIENT_SECRET)
- Configuraciones de correo para recuperación de contraseñas
- Rutas de archivos estáticos y multimedia
- Indicadores de modo de depuración (DEBUG = False en producción)

El uso de variables evita exponer información sensible en el código fuente.

Configuración de Base de Datos en Producción

El sistema fue diseñado para funcionar con MySQL, por lo que se consideró lo siguiente para su ejecución en producción:

- Activación del motor MySQL en el servidor.
- Creación de una base de datos dedicada al sistema.
- Creación de usuario y contraseña exclusivos.
- Aplicación de migraciones mediante python manage.py migrate.
- Configuración de respaldos automáticos si el proveedor lo permite.

Gestión de Archivos Estáticos y Multimedia

Para un entorno de producción se configuraron:

- Recolección de archivos estáticos mediante `python manage.py collectstatic`.
- Directorio dedicado para almacenar archivos subidos a `/media/`.
- Protección adecuada mediante servidor web (Nginx o Apache).

Ajustes de Seguridad

Se implementaron prácticas esenciales como:

- Deshabilitar DEBUG.
- Definir dominios permitidos (ALLOWED_HOSTS).
- Activar el uso de HTTPS (cuando el dominio esté configurado).
- Configuración de seguridad de cookies:
 - `CSRF_COOKIE_SECURE = True`
 - `SESSION_COOKIE_SECURE = True`

Estos ajustes garantizan que el sistema no exponga riesgos durante la operación real.

4.8.2. Implementación en Servidor o Hosting

Aunque el despliegue aún no se ha realizado, se definieron las opciones viables y los pasos necesarios para publicar el sistema. Las plataformas recomendadas para su implementación incluyen:

Render

Render es uno de los servicios más utilizados para desplegar aplicaciones Django. Permite:

- Despliegue mediante repositorio GitHub
- Configuración de variables de entorno
- Base de datos MySQL o PostgreSQL
- Certificados SSL automáticos

Pasos principales:

1. Conectar el repositorio del proyecto a Render.
2. Configurar comandos de inicio y dependencias.
3. Configurar variables de entorno.
4. Activar dominio y certificados.

Railway

Railway es otra alternativa moderna que ofrece:

- Contenedores automáticos
- Bases de datos gestionadas
- Integración con GitHub
- Consola interactiva

Pasos principales:

1. Crear un proyecto en Railway.
2. Seleccionar plantilla Django o configurar manualmente.
3. Agregar servicio de MySQL.
4. Asociar el repositorio del proyecto.
5. Configurar build y variables de entorno.

VPS (Servidor privado virtual)

Alternativa más profesional que ofrece control absoluto del entorno.

Puede utilizarse un VPS con:

- Ubuntu Server
- Nginx como servidor web
- Gunicorn como servidor WSGI
- Certbot para certificados SSL

Pasos principales:

1. Instalar Python, pip y MySQL.

2. Configurar Nginx + Unicorn.
3. Publicar el proyecto.
4. Configurar firewall.
5. Asociar dominio y activar HTTPS.

Hosting compartido (última opción)

Solo recomendable si no se requiere escalabilidad. No es ideal para Django, pero algunos hostings permiten Python mediante contenedores.

4.8.3. Mantenimiento y Actualizaciones Futuras

Una vez que el sistema sea desplegado, deberá contar con un plan de mantenimiento que garantice la estabilidad, seguridad y disponibilidad. Las actividades recomendadas son:

Respaldos automáticos

- Copias programadas de base de datos.
- Respaldos periódicos de archivos multimedia.
- Descarga mensual de información para conservación externa.

Actualización de la plataforma

El sistema podrá requerir ajustes futuros en:

- Nuevos servicios o productos ofrecidos por la empresa.
- Actualización de la pasarela de pago PayPal.
- Optimización de seguridad de Django.
- Corrección de errores o mejoras solicitadas por el cliente.

Monitoreo

Una vez en producción será necesario:

- Monitorear logs del servidor.
- Revisar intentos fallidos de acceso.
- Verificar cargas de archivos válidos.
- Supervisar rendimiento en horas de mayor tráfico.

Escalabilidad

Si la empresa incrementa la demanda, se puede:

- Migrar la base de datos a un servidor dedicado.
- Activar un CDN para acelerar recursos estáticos.
- Optimizar consultas mediante caching.
- Escalar horizontalmente con instancias adicionales.

4.9. Resultados del Desarrollo

El desarrollo del sistema permitió construir una plataforma web integral que responde directamente a las necesidades funcionales detectadas durante el análisis del proyecto. A lo largo del proceso se implementaron módulos que abarcan tanto la operación de la empresa como la experiencia del usuario final, generando una herramienta sólida, moderna y adaptable para la gestión de servicios, reservaciones y administración interna.

Este apartado presenta de manera detallada los principales resultados alcanzados, su impacto funcional y organizacional, así como las limitaciones detectadas y el valor agregado que aporta la solución.

4.9.1. Funcionalidades Implementadas

El sistema se desarrolló de forma modular, permitiendo integrar una amplia variedad de funcionalidades tanto para el sitio público como para los paneles administrativos. Entre las principales funciones implementadas se encuentran:

Panel Administrativo Interno (Empresa)

- Gestión de clientes y empleados.
- Gestión de servicios y configuraciones empresariales.
- Administración del ciclo de vida de las reservaciones.
- Validación de comprobantes de transferencia.
- Modificación de estados de pago y reservación.
- Generación de reportes PDF utilizando librerías profesionales.
- Panel de métricas con indicadores operativos clave.
- Registro de auditorías o acciones internas.

Panel de Desarrollador

- Alta, edición y activación de empresas (sistema multiempresa).
- Configuración visual completa: logos, colores, tipografías, banners.
- Gestión avanzada de roles y permisos.
- Activación o desactivación de módulos globales: servicios, tienda/productos, sistema de reseñas.
- Administración de archivos multimedia.
- Herramientas de respaldo y exportación de datos (CSV y base de datos).

Base tecnológica

- Traductor de plantillas (Template Engine) con herencia de vistas.
- Context processors para adaptar el tema visual de la empresa activa.
- Administración segura mediante Django Admin extendido.
- Integración con MySQL para gestión optimizada de datos.

En conjunto, estas funcionalidades convierten el sistema en una plataforma completa para reservas, administración y gestión empresarial.

4.9.2. Beneficios Obtenidos

La implementación del sistema generó múltiples beneficios tanto para la operación empresarial como para la experiencia del usuario final.

Beneficios para la empresa

1. **Centralización de procesos**

Todos los registros de clientes, reservaciones, pagos y servicios se encuentran en un mismo sistema, eliminando el uso de hojas de cálculo o registros manuales.

2. **Automatización de tareas operativas**

- a. Validación automática de pagos con PayPal.
- b. Registro automático de reservaciones.
- c. Actualización automática de estados según acciones administrativas.

3. **Mayor control sobre la operación interna**

Acceso a métricas y reportes PDF que facilitan la toma de decisiones.

4. **Reducción de errores humanos**

La validación automática y los formularios controlados disminuyen inconsistencias en los datos.

5. **Escalabilidad por diseño**

El sistema puede crecer sin necesidad de rehacer su estructura.

Beneficios para los clientes

1. **Experiencia de usuario clara e intuitiva**

El flujo de reservación está diseñado para ser simple, secuencial y sin confusiones.

2. **Métodos de pago flexibles**

Los usuarios pueden elegir entre pagar con tarjeta o mediante transferencia bancaria, según su preferencia.

3. **Autonomía a través del panel personal**

Los clientes pueden revisar sus reservaciones, modificar datos o cancelar servicios sin depender de la empresa.

4. **Seguridad y confianza**

La integración con PayPal refuerza la percepción de seguridad en los pagos electrónicos.

4.9.3. Limitaciones Detectadas

Si bien el sistema cumple con todos los requerimientos planteados, durante el desarrollo se identificaron algunas limitaciones y áreas que podrían mejorarse en versiones futuras:

Dependencia de conexión estable para pagos

El proceso de pago mediante PayPal requiere conexión constante; interrupciones pueden causar transacciones fallidas.

Validación manual de comprobantes

El proceso de revisión de comprobantes por transferencia depende del personal administrativo, lo que puede generar retrasos en momentos de alta demanda.

Complejidad de despliegue en VPS

El despliegue en servidores privados requiere configuración avanzada (Nginx, Unicorn, certificados SSL), lo que demanda conocimiento técnico adicional.

Falta de notificaciones automáticas

Aunque el flujo principal funciona correctamente, aún no se integraron notificaciones automáticas por correo en ciertos eventos como:

- Aprobación o rechazo de comprobantes
- Confirmación de reservación
- Actualización de estado de pago

Optimización de panel en dispositivos móviles

El panel administrativo está optimizado principalmente para escritorio; aunque es usable en móvil, puede mejorarse la experiencia en pantallas pequeñas.

Estas limitaciones no comprometen la funcionalidad central del sistema, pero representan oportunidades de mejora que podrían agregarse en fases futuras.

4.9.4. Valor Agregado al Entorno Empresarial

El desarrollo del sistema aporta un valor significativo tanto a nivel operativo como estratégico. Entre los aspectos más relevantes destacan:

Digitalización completa del proceso de reservaciones

La empresa pasa de un proceso manual a uno totalmente automatizado, lo que mejora la eficiencia y reduce errores.

Solución adaptable a múltiples empresas

El sistema no está limitado a una empresa; su estructura multiempresa permite comercializar o licenciar la plataforma a otros negocios del mismo sector o sectores relacionados.

Profesionalización de la gestión interna

El acceso a métricas, reportes PDF y auditorías transforma la operación en un proceso ordenado y medible.

Mejor experiencia del cliente

Una interfaz clara, pagos seguros y un panel personal fortalecen la confianza del usuario.

Base tecnológica escalable

La arquitectura del proyecto permite implementar módulos avanzados como:

- Chat de soporte
- Notificaciones push
- Sistema interno de mensajería
- Reservaciones con calendario en tiempo real
- Integración con APIs externas

Reutilización del sistema para nuevos contextos

El diseño modular hace posible que esta plataforma pueda utilizarse en:

- Empresas de turismo
- Hoteles o cabañas
- Spas
- Consultorios
- Negocios de renta de espacios
- E-commerce

El valor agregado radica en que no se trata solo de un sistema desarrollado para una empresa específica, sino de una **plataforma adaptable, escalable y profesional** que puede evolucionar hacia un producto comercial.

4.10. Conclusiones Parciales

El desarrollo del sistema permitió materializar una plataforma web integral, modular y escalable que responde a las necesidades identificadas durante el análisis inicial. A través de la implementación de funcionalidades orientadas tanto al usuario final como al personal administrativo y al desarrollador, se logró construir una solución completa para la gestión de servicios, reservaciones y operaciones internas empresariales.

El proceso de desarrollo evidenció la importancia de una arquitectura bien estructurada, basada en principios sólidos como modularidad, separación de responsabilidades y uso de patrones establecidos, tales como el Modelo-Vista-

Plantilla propio de Django. Esta arquitectura facilitó que el sistema pudiera evolucionar de manera organizada hacia una solución multiempresa, capaz de adaptarse a distintos giros comerciales.

Del mismo modo, la integración de herramientas externas como la pasarela de pago PayPal, el método de pago por transferencia bancaria y librerías para generación de reportes PDF contribuyó a enriquecer la funcionalidad del sistema, ampliando su alcance y consolidando una experiencia más completa para el usuario.

La implementación de los paneles administrativos —tanto el interno como el del desarrollador— permitió atender necesidades operativas reales, como la administración de servicios, empleados, clientes, pagos, reservaciones, temas visuales, contenidos institucionales y configuraciones avanzadas. Estos módulos proporcionan a la empresa un mayor control y organización, al mismo tiempo que sientan las bases para escalar la plataforma hacia nuevas funcionalidades o modelos de negocio.

Con respecto a la fase de pruebas, los resultados permitieron validar que la solución funciona correctamente en sus principales componentes, identificando además áreas de mejora que, si bien no afectan la operación actual, representan oportunidades de crecimiento futuro, como la optimización del panel administrativo para dispositivos móviles o la implementación de notificaciones automáticas por correo.

Como conclusión parcial, se considera que el sistema desarrollado cumple con los objetivos planteados inicialmente y resuelve la problemática identificada en la empresa. Asimismo, presenta una base tecnológica sólida que garantiza su mantenimiento y evolución, destacando por su flexibilidad, adaptabilidad y por el valor agregado que ofrece tanto al entorno empresarial como al proceso formativo del residente.

Estas conclusiones parciales permiten cerrar de manera integral el proceso de desarrollo expuesto en este capítulo y abren paso hacia la elaboración del capítulo

final, donde se presentarán las conclusiones generales del proyecto y las recomendaciones para trabajos futuros.

5.Conclusiones

5.1. Evaluación de Objetivos

5.1.1. Introducción

Esta sección presenta una evaluación del cumplimiento de los objetivos planteados al inicio del proyecto. Cada objetivo específico se compara con la evidencia de su implementación en el sistema desarrollado.

5.1.2. Objetivos del Proyecto

Objetivo General

Desarrollar un Sistema Integral de Gestión Operativa Web que permita a empresas del sector turístico gestionar de manera eficiente sus operaciones comerciales, servicios y reservaciones a través de una plataforma web completa y personalizable.

Tabla 18 Evaluación de Objetivos Específicos

Objetivo Específico	Evidencia de Cumplimiento	Estado
1. Implementar sistema de registro y autenticación de usuarios	Sistema de registro funcional en /registro con validación de campos, creación de usuarios con empresa asociada, y sistema de login/logout con sesiones seguras. Implementado en usuarios/views.py y usuarios/forms.py.	Cumplido
2. Desarrollar panel de administración con dashboard y métricas	Dashboard implementado en /adminpanel/ con estadísticas en tiempo real (ventas, pedidos, usuarios, reservaciones), gráficos visuales, y accesos rápidos. Vista dashboard() en adminpanel/views.py con métricas activas.	Cumplido
3. Implementar módulo de gestión de	Sistema CRUD completo de productos en /adminpanel/productos/ con creación, edición, eliminación, listado con búsqueda y filtros,	Cumplido

productos (CRUD completo)	categorías, control de stock, y atributos personalizados. Implementado en adminpanel/views.py y adminpanel/forms.py.	
4. Desarrollar sistema de e-commerce con carrito de compras y checkout	Carrito de compras en sesión implementado, vista de productos, detalle de producto, agregar/eliminar del carrito, proceso de checkout completo con métodos de pago, y generación de pedidos. Rutas en home/urls.py y vistas en home/views.py.	Cumplido
5. Implementar gestión de pedidos con estados y seguimiento	Modelo Pedido con estados (Pendiente, Confirmado, En camino, Entregado, Cancelado), vista de detalle de pedidos, actualización de estados, y generación de número de pedido único. Implementado en adminpanel/models.py y adminpanel/views.py.	Cumplido
6. Desarrollar módulo de servicios turísticos con reservaciones	Sistema de servicios con tipos (avistamiento, hospedaje, tours), gestión de reservaciones con fechas, cantidad de personas, cálculo automático de costos, y estados de reservación. Modelos en servicios/models.py y reservaciones/models.py.	Cumplido
7. Implementar gestión dinámica de servicios desde panel admin	CRUD funcional de servicios en panel admin con formularios dinámicos, atributos personalizados, imágenes, y precios configurables. Vista admin_servicios() y formularios en adminpanel/views.py.	Cumplido
8. Desarrollar sistema multi-empresa (multi-tenant)	Arquitectura multi-tenant implementada con modelo Empresa, aislamiento de datos por empresa, empresa activa en sesión, y configuración independiente por tenant. Modelo en empresas/models.py y middleware de empresa activa.	Cumplido
9. Implementar panel de desarrollo para personalización avanzada	Panel de desarrollo en /dev/ accesible solo para superusuarios, gestión de empresas, editor de temas, editor de páginas y secciones dinámicas, gestión de roles personalizados. Implementado en devpanel/views.py con múltiples vistas de gestión.	Cumplido
10. Desarrollar sistema de	Sistema de páginas (Home, Nosotros, Servicios, Productos, Galería, Contacto,	Cumplido

páginas y secciones dinámicas	Footer) con secciones configurables (Hero, Carousel, Features, Testimonios, etc.). Modelos Pagina y Seccion en devpanel/models.py con configuración JSON flexible.	
11. Implementar sistema de temas personalizables	Sistema de temas con colores configurables (primario, secundario, acento), tipografía personalizable, estilos de navbar, logos, y aplicación dinámica mediante CSS variables. Modelo Tema en empresas/models.py con formulario de edición.	Cumplido
12. Desarrollar sistema de roles y permisos dinámicos	Sistema de roles personalizados con atributos dinámicos (schema JSON), asignación de roles a usuarios, y compatibilidad con sistema legacy. Modelo Rol en devpanel/models.py con atributos_schema.	Cumplido
13. Implementar sistema de reseñas para productos y servicios	Modelo genérico Resena usando ContentType para productos y servicios, calificación de 1-5 estrellas, sistema de aprobación, y visualización en frontend. Implementado en home/models.py (o modelo genérico).	Cumplido
14. Desarrollar formulario de contacto personalizable	Formulario de contacto con campos configurables mediante JSON, almacenamiento en base de datos, envío por email vía SMTP, y gestión desde panel admin. Modelo ContactMessage con campo JSON flexible.	Cumplido
15. Implementar sistema de email con SMTP configurable	Configuración SMTP por empresa con host, puerto, usuario, contraseña, TLS/SSL, y backend de email personalizado. Configuración desde panel admin y envío de emails de confirmación.	Cumplido
16. Desarrollar exportación de reportes a PDF	Función de exportación de productos a PDF implementada usando ReportLab. Vista exportar_productos_pdf() en adminpanel/views.py con generación de documentos PDF.	Cumplido
17. Implementar atributos personalizados	Sistema de atributos personalizados mediante modelo CustomAttribute y campos JSON en productos y servicios. Permite agregar campos dinámicos según	Cumplido

para productos y servicios	necesidades del negocio. Modelo en devpanel/models.py.	
18. Desarrollar interfaz responsive y mobile-friendly	Diseño responsive implementado con Bootstrap, media queries, navbar mobile con menú hamburguesa, grid adaptativo, y optimización para dispositivos táctiles. Templates HTML con clases Bootstrap responsive.	Cumplido

5.1.3. Resumen de Cumplimiento

- **Total de objetivos:** 18
- **Objetivos cumplidos:** 18
- **Porcentaje de cumplimiento:** 100%

5.1.4. Observaciones

Todos los objetivos planteados han sido implementados exitosamente en el sistema. El desarrollo incluye:

- **Funcionalidades completas:** Todos los módulos principales están operativos
- **Arquitectura sólida:** Sistema multi-tenant bien estructurado
- **Personalización avanzada:** Temas, páginas y secciones completamente configurables
- **Experiencia de usuario:** Interfaz intuitiva tanto para clientes como para administradores
- **Escalabilidad:** Diseño que permite agregar nuevas funcionalidades fácilmente

5.1.5. Funcionalidades Adicionales Implementadas

Además de los objetivos planteados, se implementaron funcionalidades adicionales:

1. **Sistema de novedades/anuncios** para la página principal
2. **Comandos de gestión (Management Commands)** para seeders de contenido

3. **Context processors** para datos globales en templates
4. **Middleware personalizado** para control de acceso
5. **Sistema de navegación dinámica** basado en páginas activas
6. **Optimizaciones de rendimiento** con `select_related` y `prefetch_related`
7. **Validación robusta** en formularios backend y frontend

5.2. Conclusiones

El desarrollo del Sistema Integral de Gestión Operativa Web ha sido un proyecto complejo y exitoso que ha resultado en una plataforma completa y funcional para la gestión de empresas del sector turístico. A continuación, se presentan las conclusiones principales del proyecto:

5.2.1. Sobre el Proceso de Desarrollo

El proceso de desarrollo siguió una metodología ágil que permitió implementar funcionalidades de manera incremental. La arquitectura multi-tenant desde el inicio del proyecto facilitó la escalabilidad y permitió un diseño robusto que soporta múltiples empresas simultáneamente.

Fortalezas del proceso:

- Uso de Django como framework permitió un desarrollo rápido y mantenible
- Separación clara de responsabilidades entre módulos (home, adminpanel, devpanel, servicios, etc.)
- Implementación de modelos relacionales bien estructurados
- Sistema de migraciones facilitó la evolución de la base de datos

Desafíos superados:

- Implementación de sistema genérico de reseñas usando ContentType
- Gestión de contenido dinámico mediante campos JSON
- Integración de múltiples sistemas (e-commerce, reservaciones, contenido)
- Personalización avanzada sin necesidad de recompilar código

5.2.2. Sobre los Resultados Técnicos

El sistema desarrollado cumple con todos los objetivos planteados y supera las expectativas en términos de flexibilidad y personalización. La arquitectura implementada permite:

- **Gestión completa de contenido:** Las páginas y secciones son completamente configurables sin necesidad de modificar código
- **Personalización visual avanzada:** El sistema de temas permite cambiar completamente la apariencia del sitio
- **Escalabilidad:** La arquitectura multi-tenant permite agregar nuevas empresas sin afectar las existentes
- **Extensibilidad:** El sistema de atributos personalizados y roles dinámicos permite adaptarse a diferentes tipos de negocios

Tecnologías seleccionadas:

- Django 5.2 demostró ser una excelente elección por su robustez y ecosistema
- MySQL como base de datos relacional proporciona confiabilidad y rendimiento
- Bootstrap facilitó el desarrollo responsive sin necesidad de CSS complejo desde cero
- ReportLab permitió la generación de reportes PDF de manera sencilla

5.2.3. Sobre la Funcionalidad del Sistema

El sistema proporciona una solución integral que cubre:

1. **Gestión comercial:** E-commerce completo con carrito, checkout y gestión de pedidos
2. **Gestión de servicios:** Sistema de reservaciones para servicios turísticos con cálculos automáticos
3. **Contenido dinámico:** Páginas y secciones completamente configurables
4. **Administración:** Paneles intuitivos para administradores y desarrolladores

5. **Personalización:** Temas, roles, atributos personalizados según necesidades del negocio

La interfaz es intuitiva tanto para usuarios finales como para administradores, lo que facilita la adopción del sistema.

5.2.4. Sobre el Impacto y Aplicabilidad

El sistema desarrollado tiene un amplio potencial de aplicación en:

- Empresas de turismo rural y ecoturismo
- Hoteles y hospedajes
- Agencias de viajes locales
- Empresas que combinan productos y servicios
- Cualquier negocio que requiera gestión multi-sitio

La arquitectura multi-tenant permite que una sola instalación del sistema pueda servir a múltiples empresas, lo que lo hace especialmente valioso para proveedores de software como servicio (SaaS).

5.3. Recomendaciones

Basadas en la experiencia del desarrollo y el análisis del sistema, se presentan las siguientes recomendaciones para mejoras futuras y optimizaciones:

5.3.1. Recomendaciones de Corto Plazo (1-3 meses)

1. Implementar sistema de caché:

- a. Integrar Redis o Memcached para mejorar el rendimiento
- b. Cachear consultas frecuentes como configuraciones de empresa y temas
- c. Cachear resultados de páginas estáticas

2. Agregar tests automatizados:

- a. Implementar tests unitarios para modelos críticos
- b. Tests de integración para flujos principales (checkout, reservaciones)

- c. Tests de formularios y validaciones
- d. Configurar CI/CD para ejecutar tests automáticamente

3. Mejorar seguridad:

- a. Implementar rate limiting para prevenir ataques de fuerza bruta
- b. Agregar autenticación de dos factores (2FA) para superusuarios
- c. Implementar logging de actividades sensibles
- d. Auditoría de cambios en datos críticos

4. Optimización de base de datos:

- a. Revisar y agregar índices en campos frecuentemente consultados
- b. Optimizar queries N+1 con `select_related` y `prefetch_related`
- c. Implementar paginación eficiente en todas las listas

5. Mejoras en SEO:

- a. Implementar meta tags dinámicos para cada página
- b. Generar sitemap.xml automáticamente
- c. Agregar datos estructurados (Schema.org)
- d. URLs amigables y optimizadas

5.3.2. Recomendaciones de Mediano Plazo (3-6 meses)

1. API REST:

- a. Desarrollar API REST completa usando Django REST Framework
- b. Documentación con Swagger/OpenAPI
- c. Autenticación por tokens (JWT)
- d. Versión de API para compatibilidad futura

2. Sistema de notificaciones:

- a. Notificaciones push para navegadores
- b. Notificaciones por email mejoradas con templates HTML
- c. Notificaciones SMS para eventos críticos
- d. Panel de preferencias de notificaciones por usuario

3. Integración de pagos:

- a. Integración completa de PayPal
- b. Integración de pasarelas de pago locales (Stripe, Mercado Pago)

- c. Procesamiento de pagos en línea
- d. Gestión de reembolsos y cancelaciones

4. Analytics y reportes avanzados:

- a. Dashboard con gráficos interactivos (Chart.js, D3.js)
- b. Reportes personalizables por administrador
- c. Exportación a múltiples formatos (PDF, Excel, CSV)
- d. Análisis de comportamiento de usuarios

5. Sistema de cupones y descuentos:

- a. Generación de códigos de descuento
- b. Descuentos por porcentaje o cantidad fija
- c. Descuentos por categoría o producto específico
- d. Descuentos por volumen de compra

6. Mejoras en gestión de inventario:

- a. Alertas de stock bajo
- b. Historial de movimientos de inventario
- c. Gestión de proveedores
- d. Sistema de compras y órdenes de compra

5.3.3. Recomendaciones de Largo Plazo (6-12 meses)

1. Aplicación móvil:

- a. Desarrollo de app nativa iOS y Android
- b. Sincronización con el sistema web mediante API
- c. Notificaciones push nativas
- d. Funcionalidades offline

2. Inteligencia Artificial y Machine Learning:

- a. Sistema de recomendaciones de productos/servicios
- b. Análisis predictivo de demanda
- c. Chatbot para atención al cliente
- d. Detección de fraude en transacciones

3. Marketplace multi-vendor:

- a. Permitir que múltiples vendedores ofrezcan productos

- b. Sistema de comisiones
- c. Gestión independiente de inventarios
- d. Calificaciones de vendedores

4. Integración con redes sociales:

- a. Login social (Facebook, Google)
- b. Compartir productos y servicios en redes sociales
- c. Feed de Instagram automático
- d. Integración con Facebook Pixel para analytics

5. Sistema de lealtad:

- a. Programa de puntos por compras
- b. Niveles de membresía
- c. Recompensas y beneficios exclusivos
- d. Referidos y bonificaciones

6. Internacionalización (i18n):

- a. Soporte para múltiples idiomas
- b. Conversión de monedas
- c. Formatos de fecha y número según región
- d. Traducción dinámica de contenido

5.3.4. Recomendaciones de Mantenimiento

1. Documentación continua:

- a. Mantener documentación de código actualizada
- b. Documentar decisiones arquitectónicas importantes
- c. Crear guías de troubleshooting
- d. Documentar procedimientos de despliegue

2. Monitoreo y alertas:

- a. Implementar monitoreo de servidor (CPU, RAM, disco)
- b. Alertas por errores críticos
- c. Monitoreo de rendimiento de base de datos
- d. Tracking de uptime del sistema

3. Backups automatizados:

- a. Backups diarios de base de datos
- b. Backups de archivos media
- c. Almacenamiento en ubicación remota
- d. Pruebas periódicas de restauración

4. Actualizaciones regulares:

- a. Mantener Django y dependencias actualizadas
- b. Revisar y aplicar parches de seguridad
- c. Actualizar dependencias de frontend
- d. Migrar a nuevas versiones de Python cuando sea necesario

5. Optimización continua:

- a. Revisar logs periódicamente para identificar cuellos de botella
- b. Optimizar queries lentas
- c. Comprimir imágenes automáticamente
- d. Implementar CDN para archivos estáticos

5.4.5. Recomendaciones de Mejora de Experiencia de Usuario

1. Búsqueda avanzada:

- a. Búsqueda con autocompletado
- b. Filtros avanzados para productos y servicios
- c. Búsqueda por múltiples criterios simultáneos
- d. Historial de búsquedas

2. Wishlist y favoritos:

- a. Permitir guardar productos/servicios favoritos
- b. Lista de deseos compartible
- c. Notificaciones de cambios de precio

3. Comparación de productos:

- a. Funcionalidad de comparación lado a lado
- b. Tabla comparativa de características

4. Mejoras en checkout:

- a. Guardar direcciones para uso futuro
- b. Cálculo de costos de envío en tiempo real

- c. Opción de guardar información de pago (con seguridad adecuada)

5. Perfil de usuario mejorado:

- a. Historial completo de pedidos y reservaciones
- b. Estado de pedidos en tiempo real
- c. Reordenar productos anteriores
- d. Gestión de direcciones múltiples

5.5. Reflexiones Finales

El desarrollo de este sistema ha sido una experiencia de aprendizaje valiosa que ha permitido aplicar conocimientos teóricos en un proyecto real y complejo. El sistema resultante es robusto, escalable y mantenible, cumpliendo con los objetivos iniciales y sentando las bases para futuras expansiones.

La arquitectura implementada demuestra que es posible crear sistemas flexibles y personalizables sin sacrificar rendimiento o mantenibilidad. El uso de Django como framework y MySQL como base de datos ha demostrado ser una combinación efectiva para este tipo de aplicaciones empresariales.

5.5.1. Aspectos destacables del proyecto

- Arquitectura bien diseñada que facilita el mantenimiento
- Código organizado y modular
- Funcionalidades completas y probadas
- Documentación técnica completa
- Interfaz intuitiva para usuarios finales

5.5.2. Lecciones aprendidas

- La planificación de arquitectura multi-tenant desde el inicio es crucial
- Los campos JSON proporcionan flexibilidad, pero requieren validación cuidadosa
- La separación de concerns (panel admin vs devpanel) mejora la usabilidad
- La documentación durante el desarrollo facilita el mantenimiento futuro

El sistema está listo para ser desplegado en un entorno de producción con las configuraciones de seguridad apropiadas y puede servir como base sólida para futuras mejoras y expansiones.

Referencias bibliográficas

Instituto Nacional de Estadística y Geografía. (2020, junio 25). *Estadísticas a propósito del Día de las Micro, Pequeñas y Medianas Empresas (MIPyMES) – Comunicado de prensa núm. 285/20*.
<https://www.inegi.org.mx/contenidos/saladeprensa/aproposito/2020/MYPIMES20.pdf>

Laudon, K. C., & Laudon, J. P. (2016). *Management Information Systems: Managing the Digital Firm* (14.^a ed.). Pearson.

Codd, E. F. (1970). *A relational model of data for large shared data banks*. *Communications of the ACM*, 13(6), 377–387.
<https://doi.org/10.1145/362384.362685>

Instituto Nacional de Estadística y Geografía (INEGI). (2020, junio 25). *Estadísticas a propósito del Día de las Micro, Pequeñas y Medianas Empresas (MIPyMES) – Comunicado de prensa núm. 285/20*.
<https://www.inegi.org.mx/contenidos/saladeprensa/aproposito/2020/MYPIMES20.pdf>

Django Software Foundation. (2024). *Django documentation (versión 5.0)*.
<https://docs.djangoproject.com/en/5.0/>

Bootstrap Team. (2024). *Bootstrap 5 documentation*.
<https://getbootstrap.com/docs/5.0/getting-started/introduction/>

World Wide Web Consortium (W3C). (2018). *Web Content Accessibility Guidelines (WCAG) 2.1*. <https://www.w3.org/TR/WCAG21/>

PayPal Developer. (2024). *PayPal REST API documentation*.
<https://developer.paypal.com/docs/api/overview/>

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The definitive guide to Scrum – The rules of the game*. Scrum.org. <https://scrumguides.org/>

Institute of Electrical and Electronics Engineers (IEEE). (2014). *IEEE Std 830-2014 – IEEE Recommended Practice for Software Requirements Specifications*. IEEE Computer Society. <https://standards.ieee.org>

IBM. (2023). *What Is SaaS Sprawl?*. IBM. <https://www.ibm.com/think/topics/saas-sprawl>

Lufti, A., Al-Khasawneh, A. L., Al-Amayreh, E., & Al-Khasawneh, A. (2022). Barriers to big data adoption in SMEs. *Management Science Letters*, *12*(7), 499-510.

Nurqamarani, N. A., Syarief, R., & Wibowo, C. (2021). Barriers to technology adoption for SMEs: A systematic literature review. *IOP Conference Series: Earth and Environmental Science*, *940*(1), 012066.

Norman, D. A. (2013). *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books.

Garrett, J. J. (2011). *The Elements of User Experience: User-Centered Design for the Web and Beyond* (2.^a ed.). New Riders.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann.

W3C. (2018). *Web Content Accessibility Guidelines (WCAG) 2.1*. <https://www.w3.org/TR/WCAG21/>

Anexo 1. Manual de Usuario- Panel Administrativo

Guía para Administradores

Acceso al Panel de Administración

1. Iniciar sesión en el sistema con credenciales de administrador
2. Hacer clic en "Panel de Administración" en el menú superior o acceder directamente a /adminpanel/
3. Se mostrará el Dashboard principal

Dashboard

El dashboard muestra un resumen general del sistema:

- **Estadísticas principales:**
 - Total de usuarios registrados
 - Total de reservaciones
 - Total de servicios activos
 - Ventas totales
 - Clientes activos
- **Gráficos y visualizaciones:**
 - Ventas por período (día, semana, mes)
 - Productos más vendidos
 - Reservaciones próximas
 - Pedidos pendientes
- **Accesos rápidos:**
 - Gestión de productos
 - Gestión de pedidos
 - Gestión de reservaciones
 - Gestión de usuarios

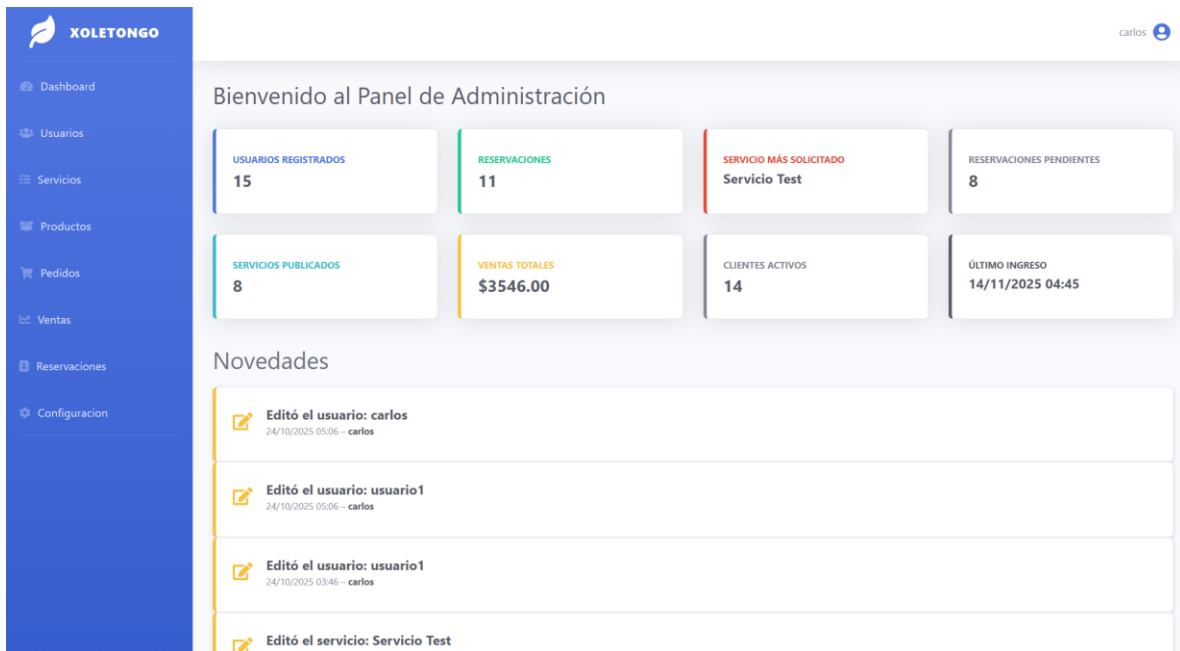


Figura 34 Dashboard panel de administración de la empresa

Gestión de Productos

Ver Lista de Productos

1. En el menú lateral, hacer clic en "Productos"
2. Se mostrará una tabla con:
 - a. Nombre del producto
 - b. Categoría
 - c. Precio
 - d. Stock disponible
 - e. Estado (Activo/Inactivo)
 - f. Acciones (Editar/Eliminar)
3. **Funciones disponibles:**
 - a. **Búsqueda:** Usar el campo de búsqueda para filtrar productos
 - b. **Filtros:** Filtrar por categoría o estado
 - c. **Paginación:** Navegar entre páginas si hay muchos productos

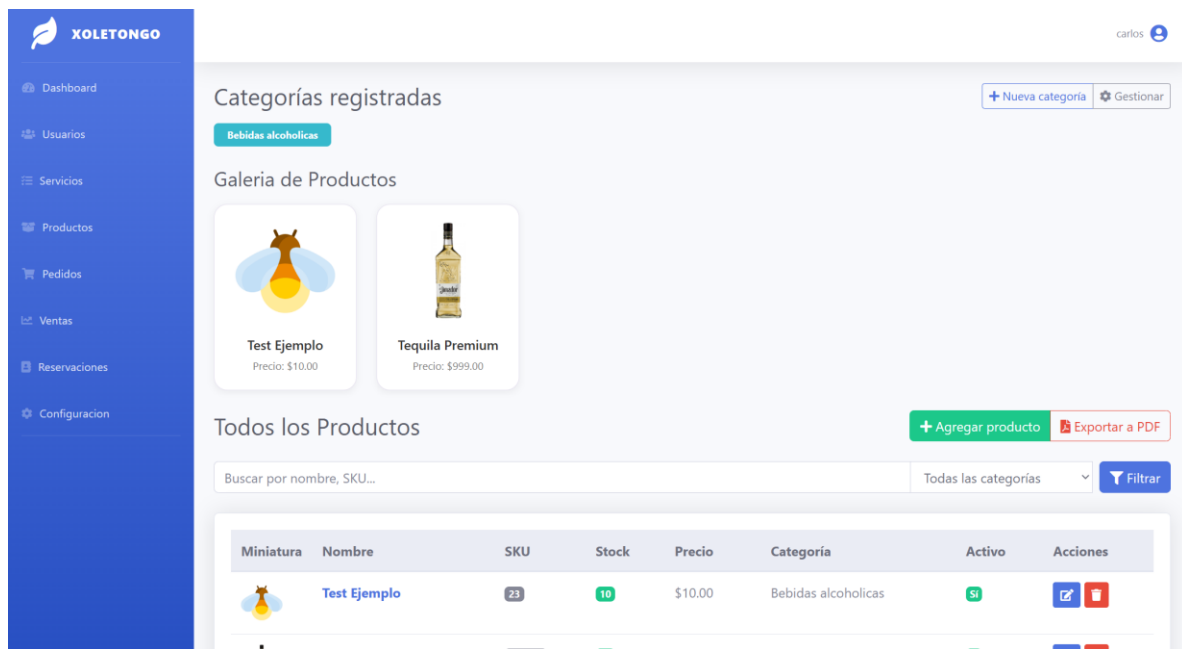


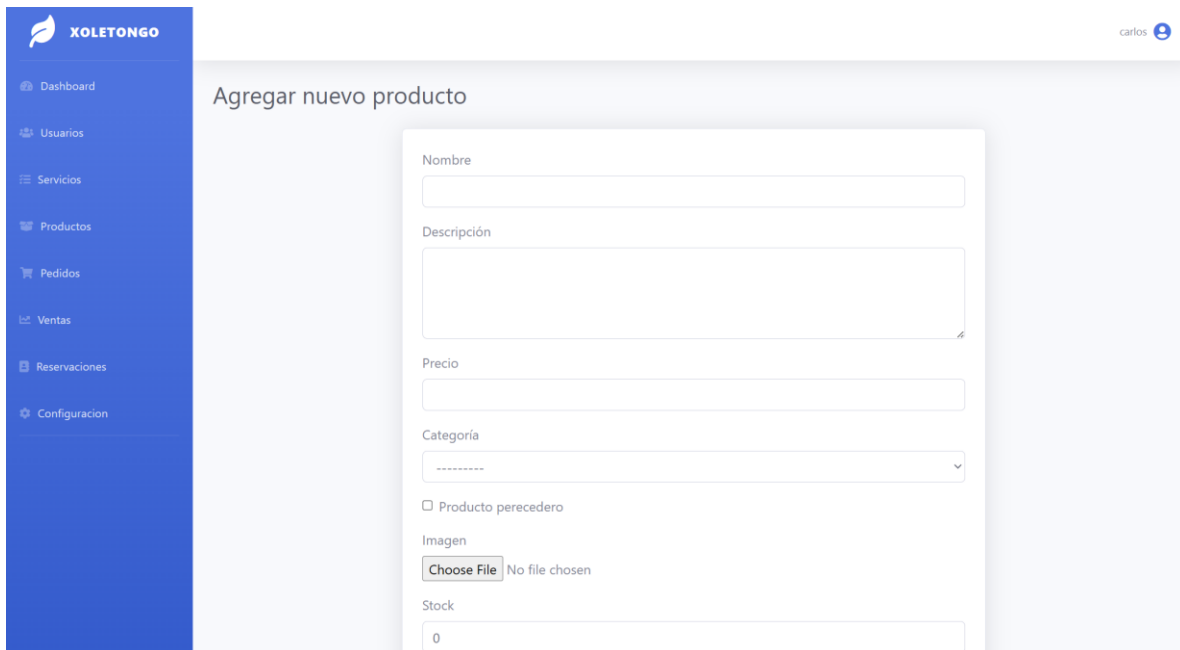
Figura 35 Gestión de productos en el administrador

Crear un Nuevo Producto

1. Hacer clic en el botón "Agregar Producto"
2. **Completar el formulario:**
 - a. **Información básica:**
 - i. Nombre del producto
 - ii. Descripción (puede usar formato HTML)
 - iii. Categoría (seleccionar o crear nueva)
 - iv. Precio
 - v. Stock inicial
 - b. **Información adicional:**
 - i. SKU (se genera automáticamente si se deja vacío)
 - ii. Imagen principal (subir archivo)
 - iii. Producto perecedero (Sí/No)
 - iv. Fecha de caducidad (si es perecedero)
 - c. **Atributos personalizados:**
 - i. Campos adicionales definidos por la empresa (talla, color, material, etc.)
 - d. **Estado:**

i. Activar/Desactivar producto

3. Hacer clic en "Guardar Producto"



The screenshot shows the 'Agregar nuevo producto' (Add new product) form in the XOLETONGO application. The form is displayed in a modal window over a light gray background. On the left, there is a blue sidebar with the XOLETONGO logo and a menu with the following items: Dashboard, Usuarios, Servicios, Productos, Pedidos, Ventas, Reservasiones, and Configuración. The top right corner of the application shows the user's name 'carlos' next to a profile icon. The form itself contains the following fields and options:

- Nombre:** A text input field.
- Descripción:** A larger text area for product details.
- Precio:** A text input field for the product price.
- Categoría:** A dropdown menu with a downward arrow.
- Producto perecedero:** A checkbox option.
- Imagen:** A file upload section with a 'Choose File' button and the text 'No file chosen'.
- Stock:** A text input field with the value '0'.

Figura 36 Formulario agregar producto

Editar un Producto

1. Desde la lista de productos, hacer clic en "Editar" junto al producto deseado
2. Modificar los campos necesarios
3. Hacer clic en "Guardar Cambios"

XOLETONGO

carlos

Editar Producto: Tequila Premium

Nombre
Tequila Premium

Descripción
Tequila de la mas alta calidad

Precio
999.00

Categoría
Bebidas alcoholicas

Imagen
Imagen actual:

Cambiar imagen:
Choose File No file chosen

Stock
10

SKU
000001

☒ Producto perecedero

Fecha de Caducidad

Figura 37 Editar producto

Eliminar un Producto

1. Desde la lista de productos, hacer clic en "Eliminar"
2. Confirmar la eliminación en el diálogo de confirmación
3. **Nota:** Se recomienda desactivar productos en lugar de eliminarlos para mantener el historial

Galeria de Productos

Test Ejemplo
Precio: \$10.00

Todos los Productos

Buscar por nombre, SKU,...

+ Agregar producto Exportar a PDF

Todas las categorías Filtrar

Confirmar Eliminación

¿Estás seguro de que quieres eliminar el producto "Tequila Premium"?

Esta acción no se puede deshacer.

Cancelar Sí, eliminar

Miniatura	Nombre	SKU	Stock	Precio	Categoría	Activo	Acciones
	Test Ejemplo	23	10	\$10.00	Bebidas alcoholicas	Si	
	Tequila Premium	000001	10	\$999.00	Bebidas alcoholicas	Si	

Figura 38 Confirma eliminación

Gestionar Categorías

1. Hacer clic en "Categorías" en el menú lateral
- 2. Crear categoría:**
 - a. Hacer clic en "Nueva Categoría"
 - b. Ingresar nombre y descripción
 - c. Guardar
3. Las categorías permiten organizar productos y facilitar la navegación

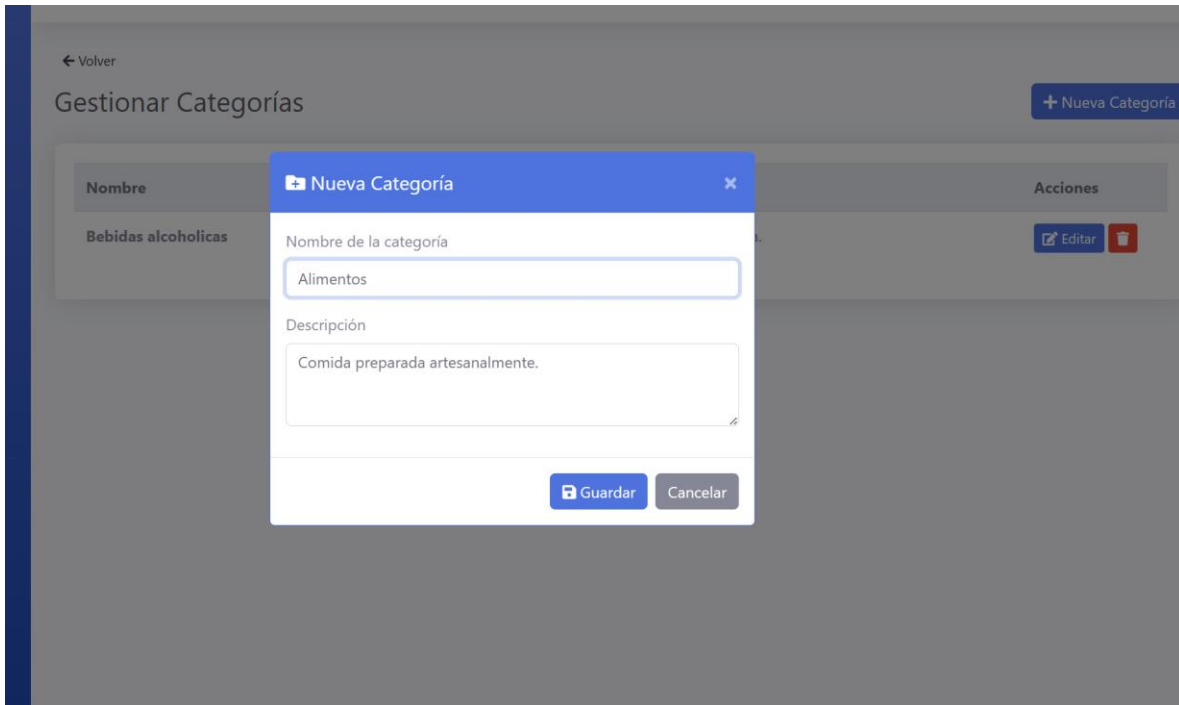


Figura 39 Agregar nueva categoría

Exportar Productos a PDF

1. Desde la lista de productos, hacer clic en "Exportar PDF"
2. Se generará un documento PDF con la lista completa de productos
3. El archivo se descargará automáticamente

Listado de Productos

Nombre	SKU	Precio	Stock	Categoría
Tequila Premium	000001	\$999.00	10	Bebidas alcoholicas
Test Ejemplo	23	\$10.00	10	Bebidas alcoholicas

Figura 40 Reporte de lista de productos en PDF

Gestión de Pedidos

Ver Lista de Pedidos

1. Hacer clic en "Pedidos" en el menú lateral
2. La tabla mostrará:
 - a. Número de pedido
 - b. Cliente
 - c. Fecha del pedido
 - d. Total
 - e. Estado (Pendiente, Confirmado, En camino, Entregado, Cancelado)
 - f. Método de pago
3. Filtros disponibles:
 - a. Por estado
 - b. Por fecha
 - c. Por método de pago
 - d. Búsqueda por número de pedido o cliente

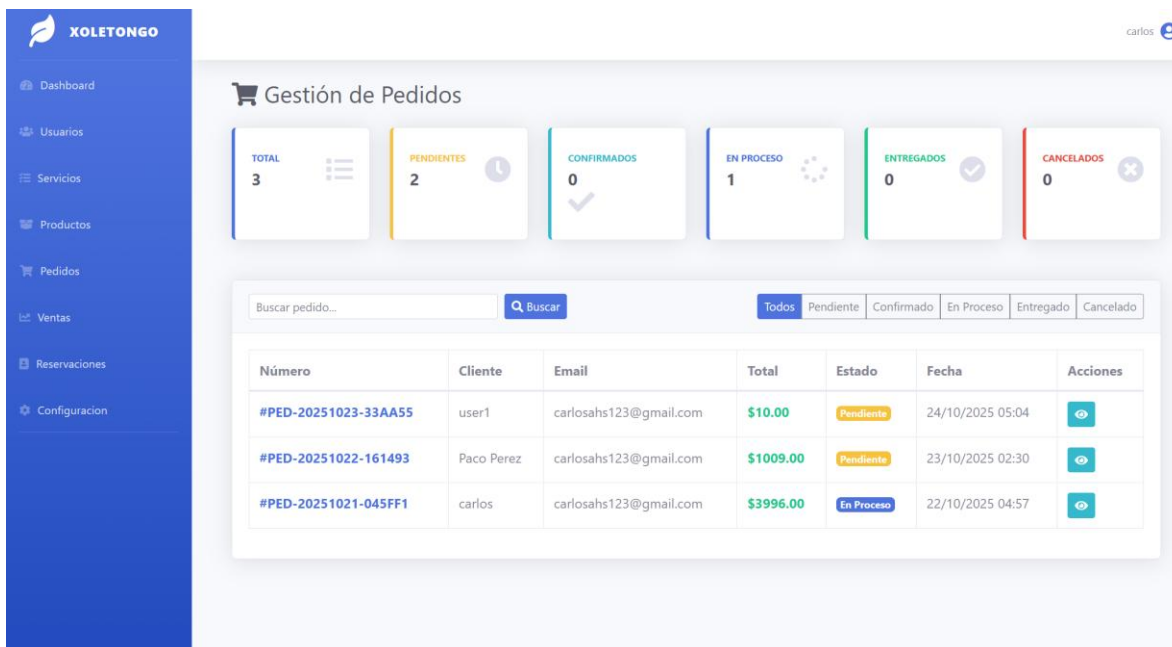


Figura 41 Gestión de pedidos

Ver Detalle de Pedido

1. Hacer clic en el número de pedido deseado

2. Se mostrará información completa:

a. Información del cliente:

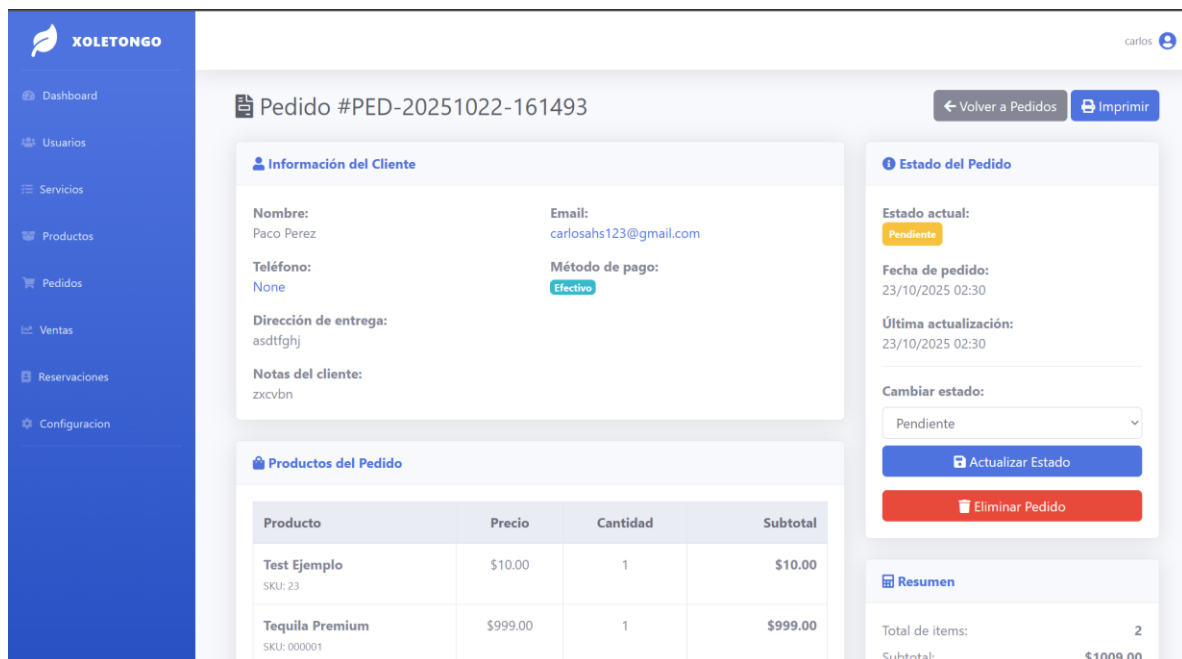
- i. Nombre completo
- ii. Correo electrónico
- iii. Teléfono
- iv. Dirección de entrega
- v. Notas del cliente

b. Productos del pedido:

- i. Lista con cantidades, precios unitarios y subtotales

c. Resumen financiero:

- i. Subtotal
- ii. Total del pedido
- iii. Método de pago seleccionado



Actualizar Estado del Pedido

Estado del Pedido

Estado actual:

Entregado

Fecha de pedido:

23/10/2025 02:30

Última actualización:

14/11/2025 05:35

Cambiar estado:

Entregado

Pendiente

Confirmado

En Proceso

Entregado

Cancelado

Figura 43 Cambiar el estado del pedido

Gestión de Reservasiones

Ver Lista de Reservasiones

1. Hacer clic en "Reservasiones" en el menú lateral
2. Se mostrará una tabla con:
 - a. Cliente
 - b. Servicio reservado
 - c. Fecha de inicio
 - d. Fecha de fin

- e. Estado
- f. Total pagado

3. Filtros:

- a. Por fecha
- b. Por servicio
- c. Por estado
- d. Búsqueda por nombre de cliente

ID	Cliente	Servicio	Tipo	Fecha(s)	Total	Pago	Estado	Acciones
#12	carlos carlosahs123@gmail.com	Habitación Doble Hacienda San Buenaventura	N/D	—	\$3,200.00	Pendiente	Pendiente	14/11/2025 05:04
#11	carlos carlosahs123@gmail.com	Servicio Test	N/D	—	\$640.00	Pendiente	Pendiente	22/10/2025 04:29
#10	carlos carlosahs123@gmail.com	Servicio Test	N/D	—	\$520.00	Pendiente	Pendiente	20/10/2025 03:59
#9	carlos carlosahs123@gmail.com	Servicio Test	N/D	—	\$520.00	Pendiente	Pendiente	20/10/2025 03:56
#5	carlos carlosahs123@gmail.com	Servicio Básico	N/D	—	\$1,000.00	Pendiente	Aprobada	18/07/2025 02:43

Figura 44 Lista y filtrado de reservaciones

Ver Detalle de Reservación

1. Hacer clic en la reservación deseada
2. Se mostrará:
 - a. Información del cliente
 - b. Detalles del servicio
 - c. Fechas y horarios
 - d. Cantidad de personas (adultos, niños, con descuento)
 - e. Total y estado de pago
 - f. Comentarios del cliente

Actualizar Estado de Reservación

1. Seleccionar el nuevo estado del menú desplegable

2. Hacer clic en "Actualizar"
3. Los cambios se guardarán automáticamente

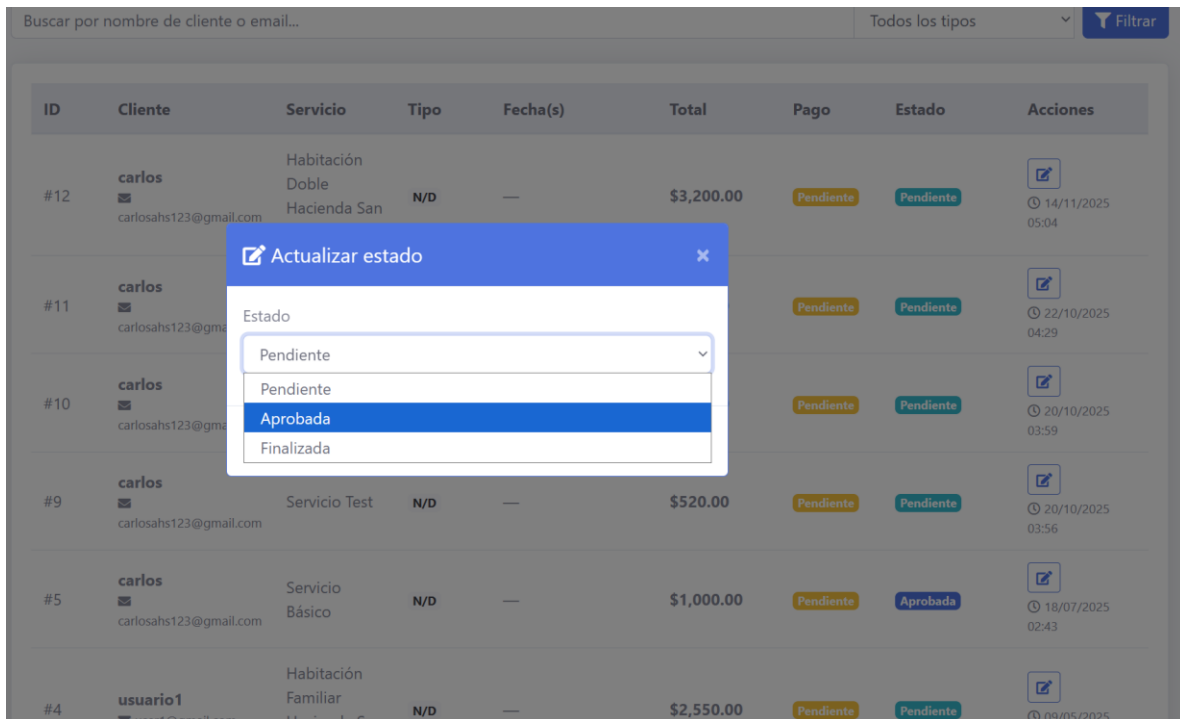


Figura 45 Actualizar estado

Gestión de Usuarios

Ver Lista de Usuarios

1. Hacer clic en "Usuarios" en el menú lateral
2. Se mostrará una tabla con:
 - a. Nombre de usuario
 - b. Correo electrónico
 - c. Nombre completo
 - d. Rol asignado
 - e. Empresa
 - f. Estado (Activo/Inactivo)

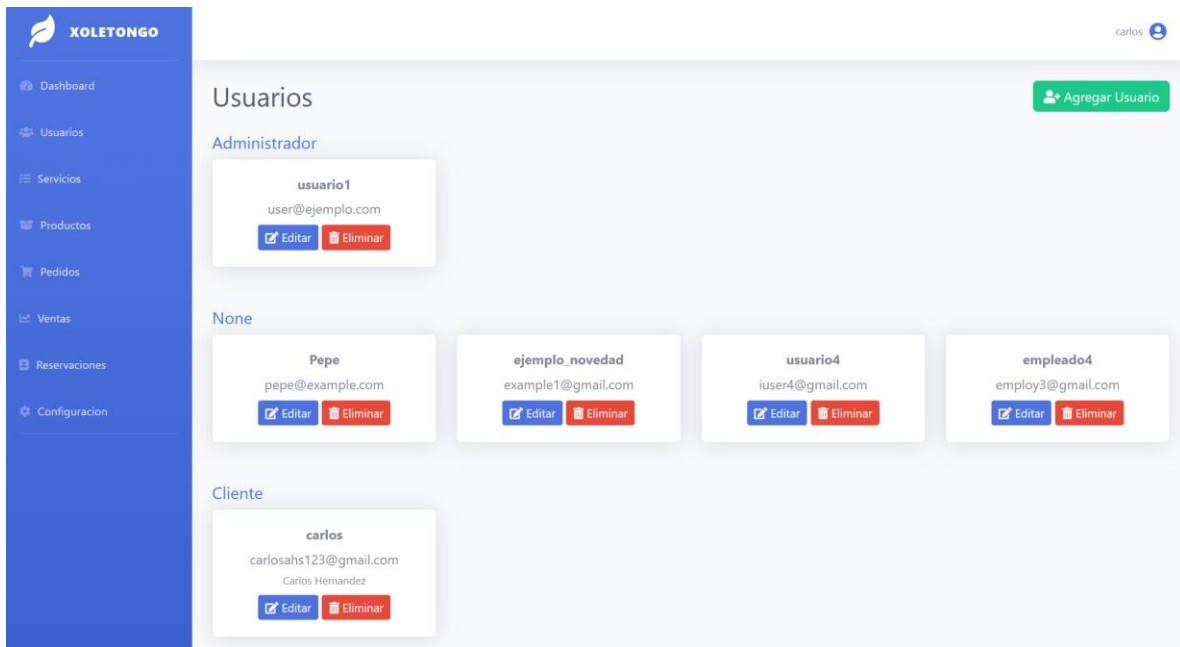


Figura 46 Gestión de usuarios

Lista completa Exportar PDF Buscar usuario...

Usuario	Email	Rol	Acciones
usuario1	user@ejemplo.com	Administrador	Editar Eliminar
Pepe	pepe@example.com	None	Editar Eliminar
ejemplo_novedad	example1@gmail.com	None	Editar Eliminar
usuario4	iuser4@gmail.com	None	Editar Eliminar
empleado4	employ3@gmail.com	None	Editar Eliminar
Danitza	dany@gmail.com	None	Editar Eliminar
admin2	admin2@gmail.com	None	Editar Eliminar
empleado3	empleado3@xoletongo.com	None	Editar Eliminar
empleado2	empleado@xoletongo.com	None	Editar Eliminar
admin	admin@gmail.com	None	Editar Eliminar

< 1 2 >

Figura 47 Lista de usuarios y exportar en PDF

Crear Usuario

1. Hacer clic en "Agregar Usuario"
2. **Completar el formulario:**

- a. Nombre de usuario (único)
- b. Correo electrónico (único)
- c. Contraseña
- d. Confirmar contraseña
- e. Rol (seleccionar del menú desplegable)
- f. Teléfono (opcional)
- g. Estado (Activo/Inactivo)

3. Si el rol tiene atributos personalizados:

- a. Aparecerán campos adicionales según la configuración del rol
- b. Completar los campos requeridos

4. Hacer clic en "Guardar Usuario"

carlos

Agregar Usuario [← Volver al listado](#)

Nombre de usuario: usuario1

Email: example1@gmail.com

Contraseña:

Confirmar contraseña:

Rol: Administrador

Teléfono: 24189081

☒ Activo

[Guardar usuario](#) [Cancelar](#)

Figura 48 Agregar nuevo usuario

Editar Usuario

1. Desde la lista, hacer clic en "Editar" junto al usuario
2. Modificar los campos necesarios (incluyendo contraseña si es necesario)
3. Hacer clic en "Guardar Cambios"

 **Editar usuario**

[← Volver al listado](#)

Nombre de usuario

empleado4

Email

employ3@gmail.com

Teléfono

2461719327

Rol

Administrador

☒ Activo

 **Atributos Personalizados del Rol: Administrador**

Completa los siguientes campos según el rol asignado.

Nivel

Senior

Departamento

1

 Guardar cambios

Cancelar

Figura 49 Editar usuario

Eliminar Usuario

1. Hacer clic en "Eliminar" junto al usuario
2. Confirmar la eliminación
3. **Advertencia:** Esta acción no se puede deshacer

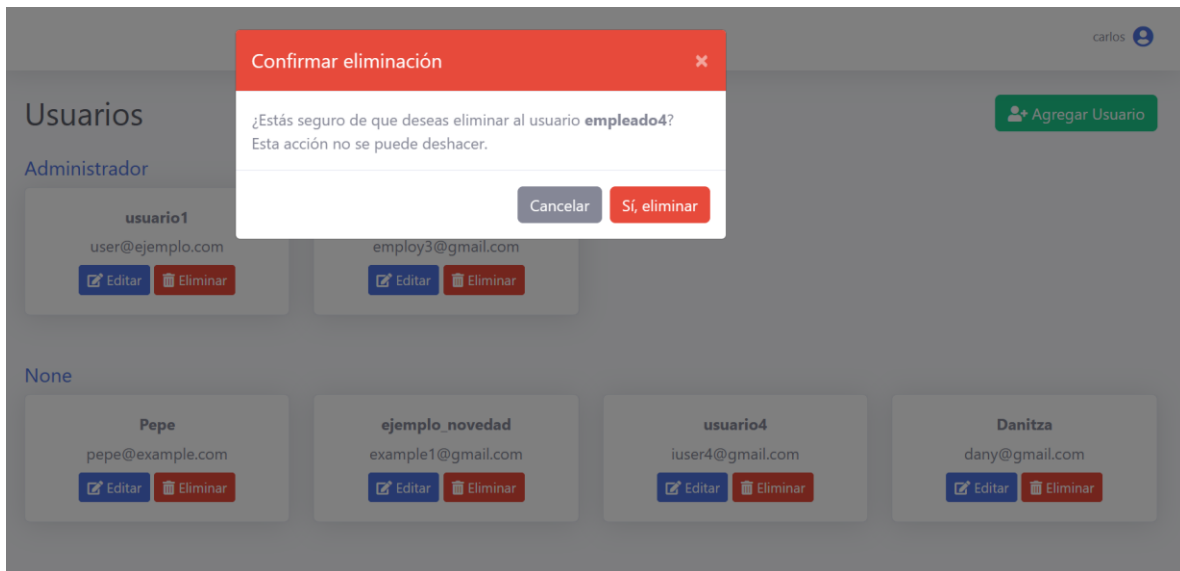


Figura 50 Eliminación de usuario

Configuración de la Empresa

Editar Información de la Empresa

1. Hacer clic en "Configuración" → "Empresa"
2. **Secciones disponibles:**

Información básica:

- a. Nombre de la empresa
- b. Nombre del titular
- c. Correo de contacto
- d. Teléfono
- e. Ubicación física

Información bancaria:

- f. Cuenta bancaria
- g. CLABE
- h. Número de tarjeta

Branding:

- i. Subir logotipo
- j. Sitio web
- k. Dominio personalizado (opcional)

Configuración SMTP (para correos):

- l. Host SMTP (ej: smtp.gmail.com)
 - m. Puerto (587 para TLS, 465 para SSL)
 - n. Usuario SMTP
 - o. Contraseña o token
 - p. Activar TLS o SSL
3. Hacer clic en "Guardar Cambios" después de cada sección

Importante: La configuración SMTP es necesaria para que funcionen los correos electrónicos de confirmación de pedidos y reservaciones.

Configuración de la empresa

Datos generales

Nombre de la empresa: Xoletongo ☒ Activa

Nombre del titular: 1 Correo de contacto: contacto@xoletongo.com

Teléfono: 5551234567 Ubicación: Santa Cruz Moxolahuá, Tlaxcala

Sitio web: Logotipo: No file chosen

Datos bancarios

Cuenta bancaria: 1234567890 CLABE: 00201012345678901 Número de tarjeta: 2

Ayuda

Solo los usuarios con rol **Administrador** pueden editar esta configuración.

Si ves los campos en gris, tienes acceso de solo lectura.

Para SMTP no actives **TLS** y **SSL** al mismo tiempo.

Figura 51 Configuración de la empresa

Configuración SMTP

Servidor SMTP: Puerto: 587 Usuario:

Contraseña: ☒ Usar TLS ☐ Usar SSL

Figura 52 Configuración SMTP de la empresa

Guía para Desarrolladores (Panel de Desarrollo)

Acceso al Panel de Desarrollo

El panel de desarrollo es accesible solo para superusuarios:

1. Iniciar sesión con credenciales de superusuario
2. Acceder a /dev/ en la URL
3. Se mostrará el menú principal del devpanel

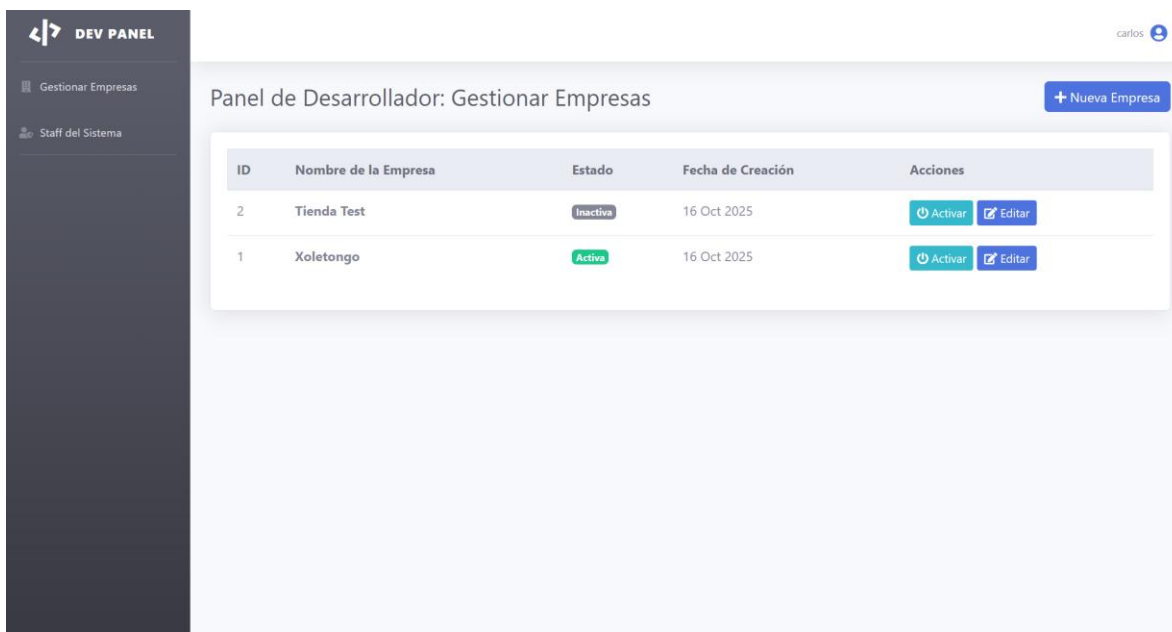


Figura 53 Panel de Desarrollador

Gestión Multi-Empresa

Crear Nueva Empresa

1. Hacer clic en "Empresas" → "Gestionar Empresas"
2. Hacer clic en "Crear Nueva Empresa"
3. **Completar información:**
 - a. Nombre de la empresa
 - b. Información de contacto
 - c. Datos bancarios
 - d. Subdominio (ej: "xoletongo")
 - e. Dominio personalizado (opcional)
4. Hacer clic en "Guardar"

5. Activar empresa:

- Marcar la casilla "Activa" para que sea la empresa visible en el sitio público
- Solo puede haber una empresa activa a la vez

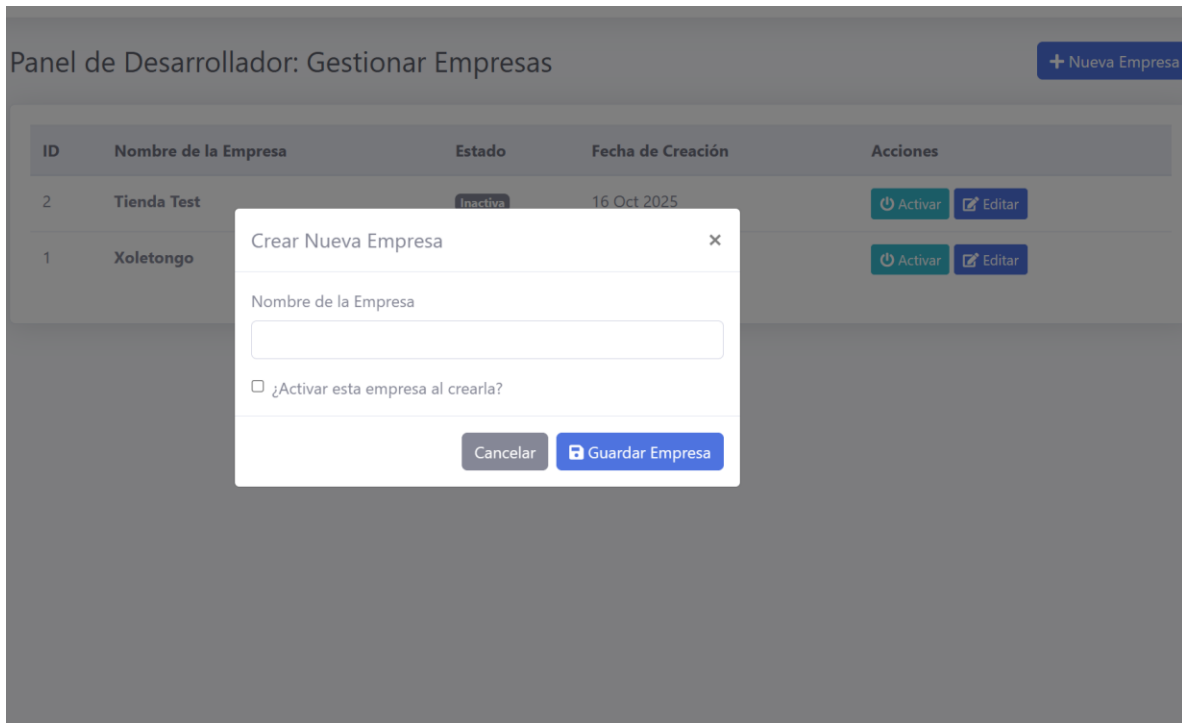


Figura 54 Agregar nueva empresa

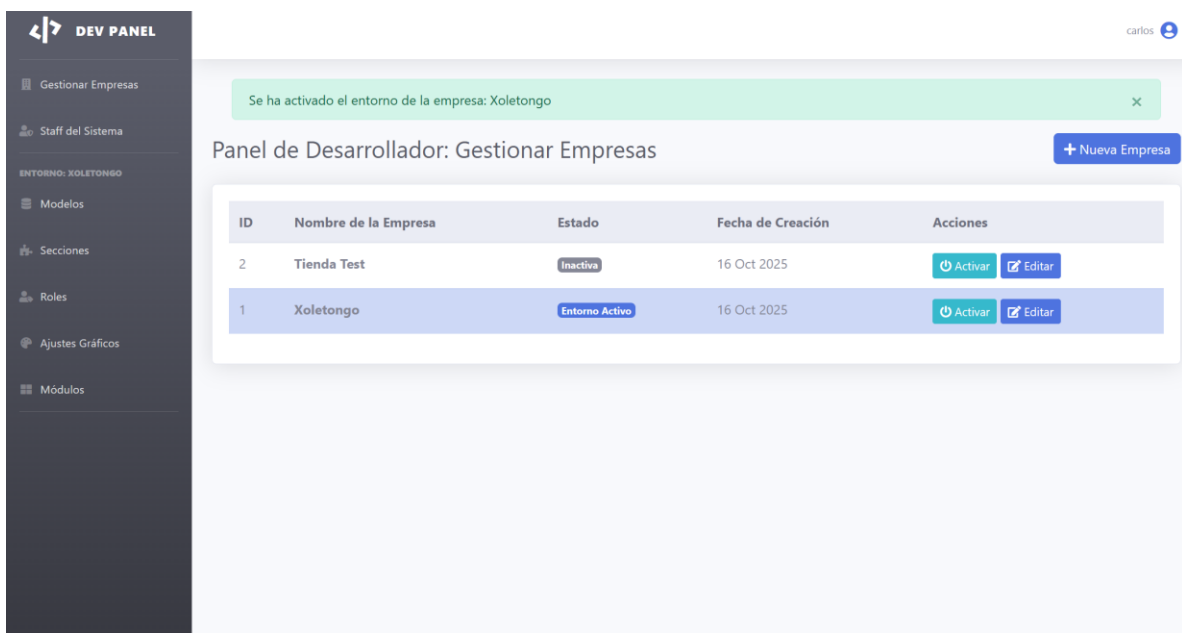


Figura 55 Activar empresa

1. Hacer clic en "Temas" → "Gestión de Temas"
- 2. Personalización de colores:**
 - a. Color primario (botones, enlaces principales)
 - b. Color secundario (hover, fondos)
 - c. Color de acento (CTAs, highlights)
 - d. Color de texto
 - e. Color de fondo
- 3. Tipografía:**
 - a. Fuente principal (body text)
 - b. Fuente secundaria (headings)
 - c. Seleccionar de Google Fonts o usar fuentes del sistema
- 4. Estilo de Navbar:**
 - a. Transparente con scroll (inicia transparente, se vuelve sólida al scrollear)
 - b. Negro sólido (siempre negro)
 - c. Color primario (usa el color principal del tema)
- 5. Imágenes:**
 - a. Logo para header
 - b. Logo para footer
 - c. Favicon
 - d. Imagen hero (opcional)
6. Hacer clic en "Guardar Tema"

DEV PANEL

Gestionar Empresas

Staff del Sistema

ENTORNO: XOLETONGO

Modelos

Secciones

Roles

Ajustes Gráficos

Módulos

carlos

Empresa: Xoletongo

Ajustes Gráficos

Personaliza la apariencia visual de tu sitio web. Los cambios se aplicarán inmediatamente.

Información del Tema

Nombre del Tema

Xoletongo Verde

Tema activo

Paleta de Colores

Selección de colores

Color Primario (navbar, títulos)

Usado en navbar, encabezados y elementos principales

Vista previa

Color Secundario

Elementos auxiliares

Vista previa

Color de Acento (botones, enlaces)

Usado en botones, enlaces y llamados a la acción

Botón CTA

Figura 56 Ajustes gráficos

Color del Texto

Color del texto general

Color de Fondo

Color de fondo del sitio

Tipografía

Fuente Principal

Roboto

Para textos generales

Fuente Secundaria

Open Sans

Para títulos y encabezados

Imágenes y Logos

Logo del Header

Choose File

No file chosen

Logo que aparece en el header de la página

Favicon (16x16 o 32x32 px)

Choose File

No file chosen

Icono que aparece en la pestaña del navegador

Imagen Hero/Banner

Choose File

No file chosen

Banner principal de la página

Figura 57 Ajustes de color, tipografía e imágenes

127

Estilo de Barra de Navegación

Elige cómo se comporta la barra de navegación en tu sitio.

☒ Transparente con scroll

☐ Siempre color primario

☐ Siempre negro

Nota: El estilo "Transparente con scroll" hace que la navbar sea transparente al inicio y cambie a un color sólido al hacer scroll. Los otros estilos mantienen el color fijo desde el principio.

Guardar Ajustes Gráficos Cancelar

Figura 58 Estilo de barra de navegación

Editor de Páginas y Secciones

Gestionar Páginas

1. Hacer clic en "Contenido" → "Gestión de Páginas"
2. Se mostrarán las páginas disponibles:
 - a. Home (Inicio)
 - b. Nosotros
 - c. Servicios
 - d. Productos
 - e. Galería
 - f. Contacto
 - g. Footer
3. **Activar/Desactivar páginas:**
 - a. Usar el toggle junto a cada página
4. **Editar título de página:**
 - a. Hacer clic en "Editar" y modificar el título

Gestionar Secciones

1. Seleccionar una página de la lista
2. Hacer clic en "Ver Secciones" o "Gestionar Secciones"
3. Se mostrará la lista de secciones actuales con su orden

Crear Nueva Sección

1. Hacer clic en "Crear Nueva Sección"

2. Seleccionar tipo de sección:

- a. **Hero/Banner:** Banner principal con imagen de fondo
- b. **Carousel:** Carrusel de imágenes
- c. **Texto con imágenes:** Bloques de texto con 1-2 imágenes
- d. **Features:** Tarjetas de características
- e. **Testimonios:** Sección de reseñas
- f. **CTA:** Botón de llamado a la acción
- g. **Galería:** Grid de imágenes
- h. **Mapa:** Google Maps embebido
- i. **Instagram Feed:** Galería de redes sociales
- j. **Formulario de contacto:** Formulario personalizable
- k. **Accordion de servicios:** Tarjetas expandibles

3. Configurar la sección:

- a. **Título:** Nombre interno de la sección
- b. **Orden:** Número que determina la posición (menor = primero)
- c. **Estado:** Activa/Inactiva

4. Configuración JSON:

- a. Se mostrará un editor JSON con campos específicos según el tipo de sección
- b. Ejemplo para Hero:

```
{
  "titulo": "Bienvenido a Xoletongo",
  "subtitulo": "Turismo Natural Sostenible",
  "bg_class": "breadcam_bg_tienda"
}
```

- c. Ejemplo para Formulario de Contacto:

```
{
  "titulo": "Contáctanos",
  "texto_boton": "Enviar Mensaje",
  "boton_bg_color": "#2d6a2e",
}
```

```

"campos":
{
  "nombre":
  "tipo":
  "placeholder":
  "requerido":
}
]
}

```

5. Hacer clic en "Guardar Sección"

Editar Sección

1. Desde la lista de secciones, hacer clic en "Editar"
2. Modificar los campos necesarios
3. Actualizar la configuración JSON si es necesario
4. Hacer clic en "Guardar Cambios"

Reordenar Secciones

1. Usar el campo "Orden" para cambiar la posición
2. Las secciones con orden menor aparecerán primero
3. Guardar los cambios

Activar/Desactivar Secciones

- Usar el toggle "Activa" para mostrar u ocultar secciones sin eliminarlas
- Útil para pruebas o contenido temporal

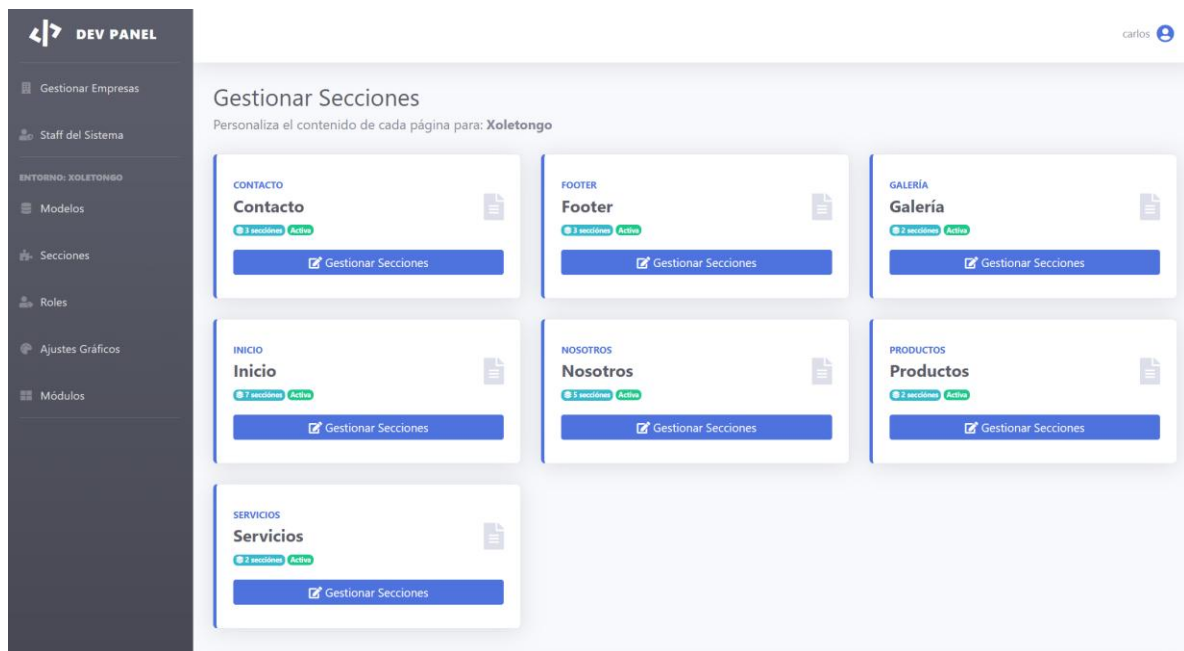


Figura 59 Gestionar secciones del sistema

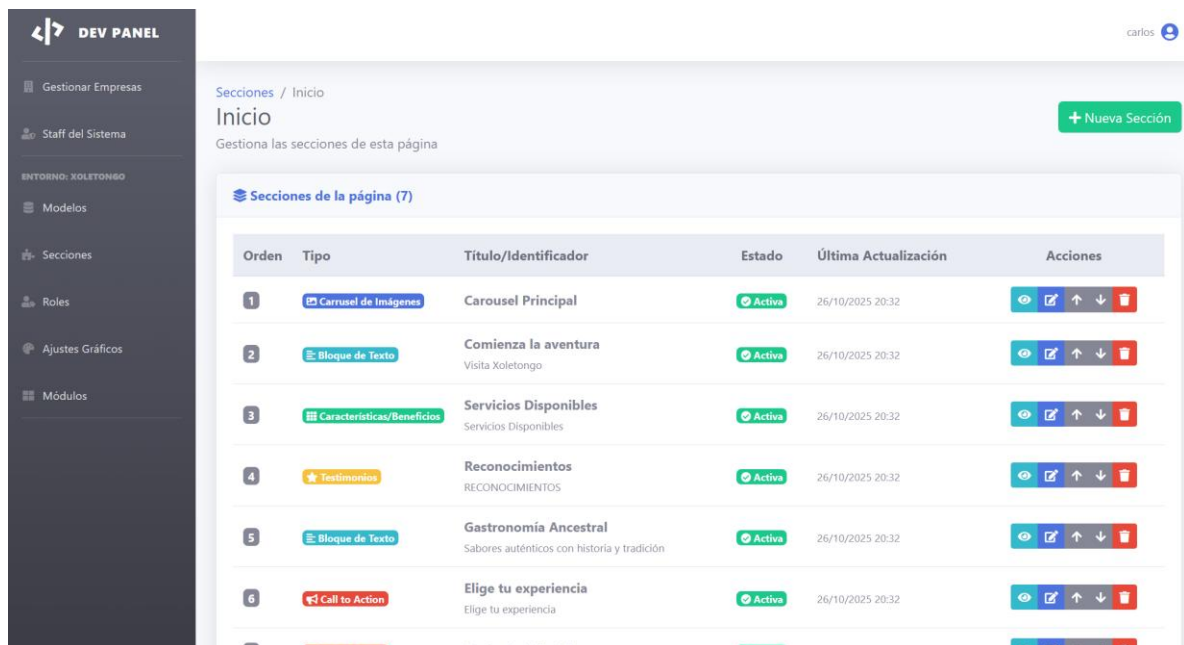


Figura 60 Secciones de la página



Figura 61 Vista previa

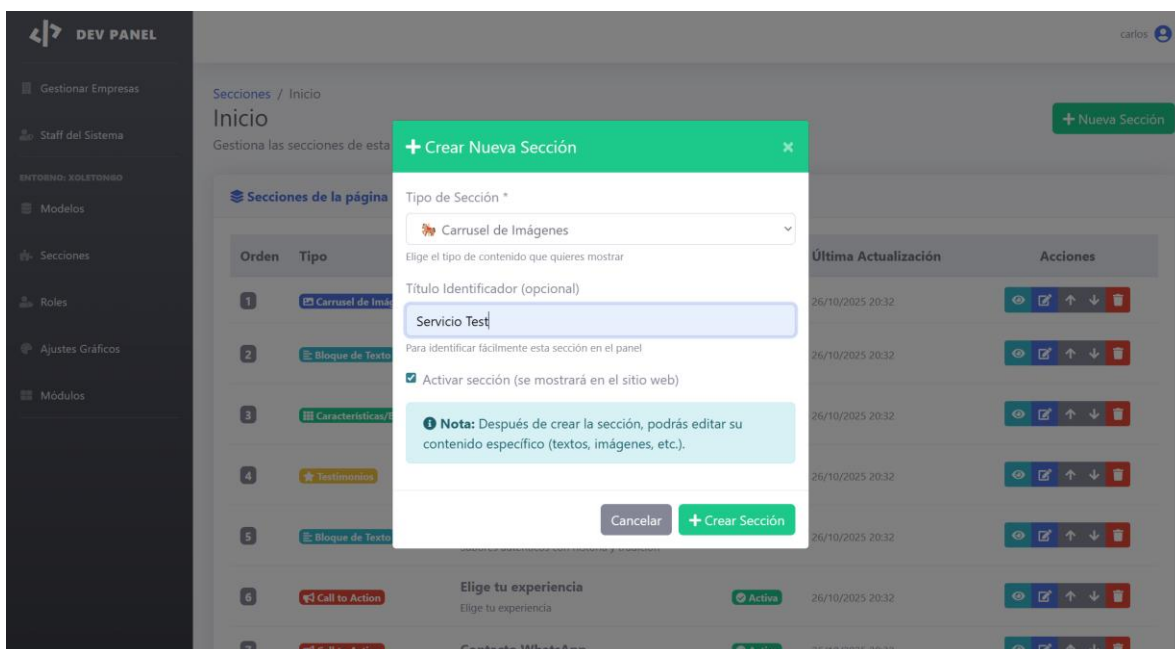


Figura 62 Crear nueva sección

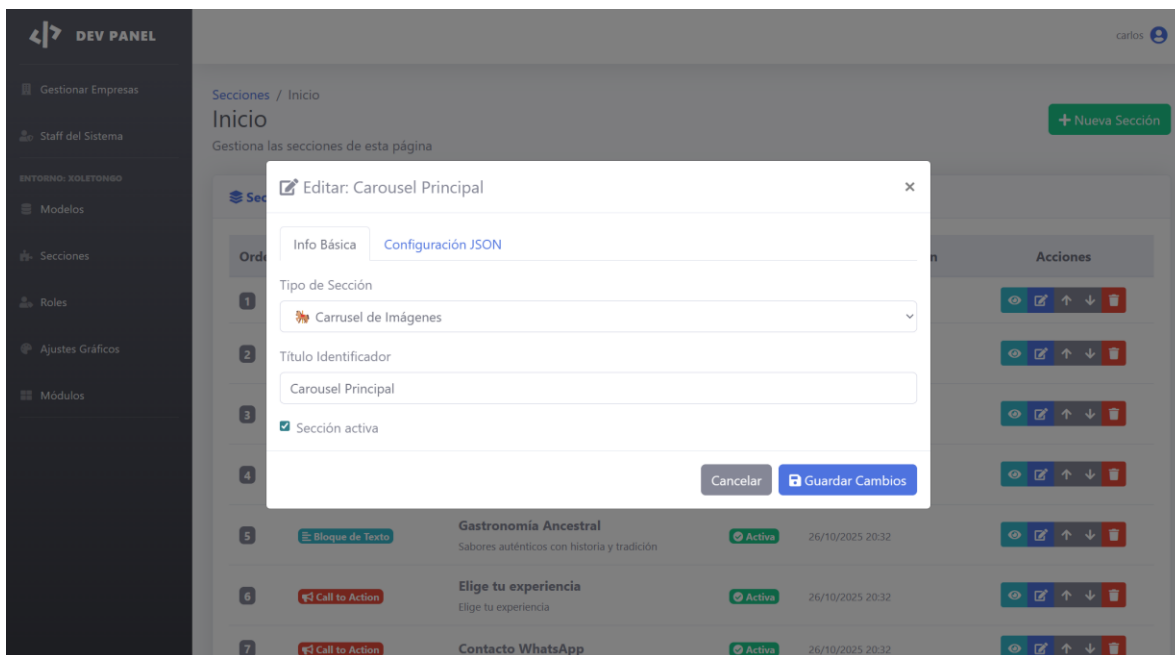


Figura 63 Editar sección

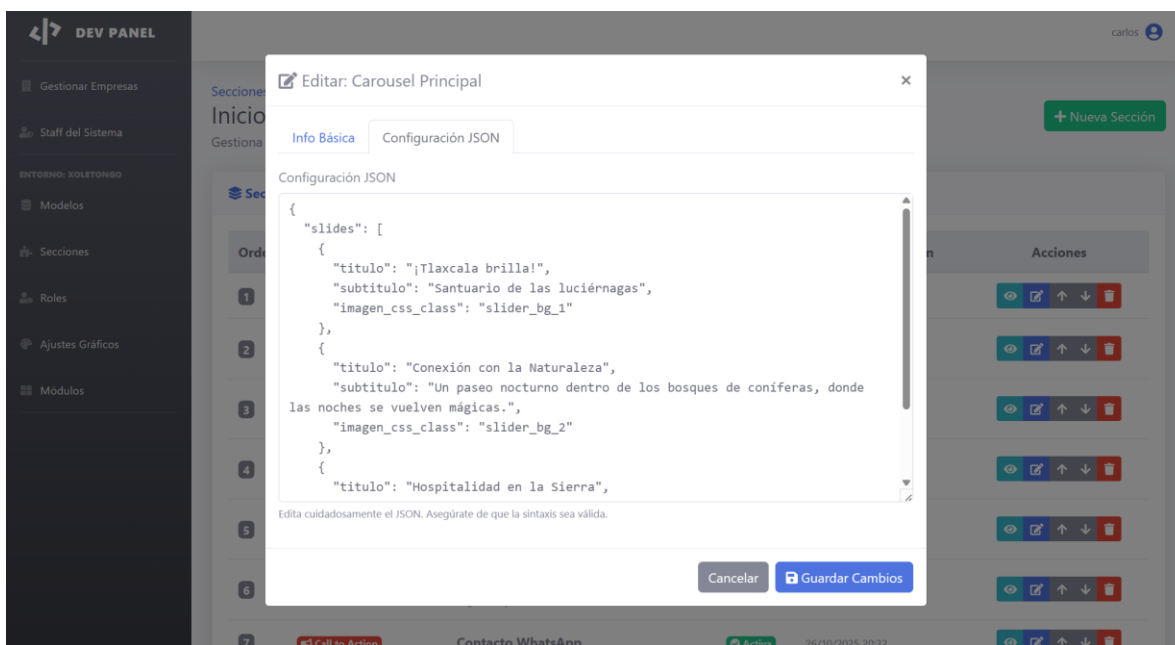


Figura 64 Configuración JSON

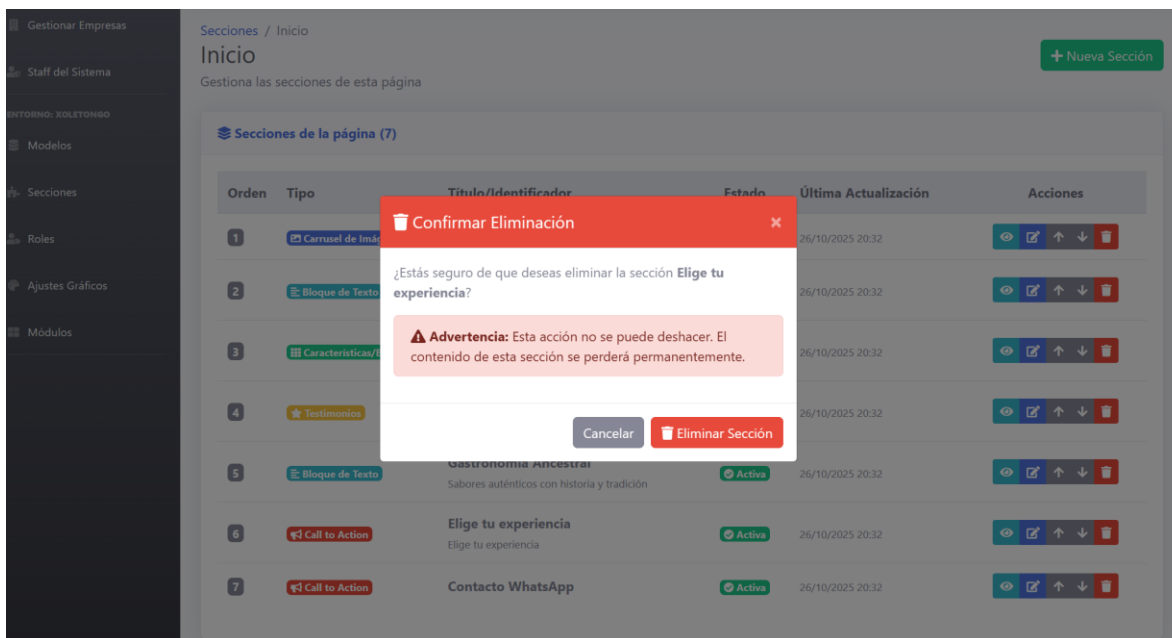


Figura 65 Eliminar sección

Sistema de Roles Dinámicos

Crear Rol Personalizado

1. Hacer clic en "Roles" → "Gestionar Roles"
2. Hacer clic en "Crear Nuevo Rol"
3. **Completar información básica:**
 - a. Nombre del rol (ej: "Guía Turístico", "Recepcionista")
 - b. Descripción
 - c. Empresa (seleccionar a qué empresa pertenece)
 - d. Estado (Activo/Inactivo)
4. **Definir atributos personalizados (Schema):**
 - a. Hacer clic en "Agregar Atributo"
 - b. Para cada atributo:
 - i. **Nombre del campo:** (ej: "Turno", "Especialidad")
 - ii. **Tipo de dato:**
 1. Texto corto
 2. Texto largo
 3. Número
 4. Fecha

5. Lista de opciones
 6. Lista múltiple
 7. Checkbox
- iii. **Requerido:** Marcar si el campo es obligatorio
 - iv. **Opciones:** (solo para listas) Separar opciones por comas

Ejemplo de schema:

```
{
  "Turno": {
    "tipo": "lista",
    "opciones": ["Matutino", "Vespertino", "Nocturno"],
    "requerido": true
  },
  "Especialidad": {
    "tipo": "texto",
    "requerido": false
  },
  "Idiomas": {
    "tipo": "lista_multiple",
    "opciones": ["Español", "Inglés", "Francés"],
    "requerido": true
  }
}
```

5. Hacer clic en "Guardar Rol"

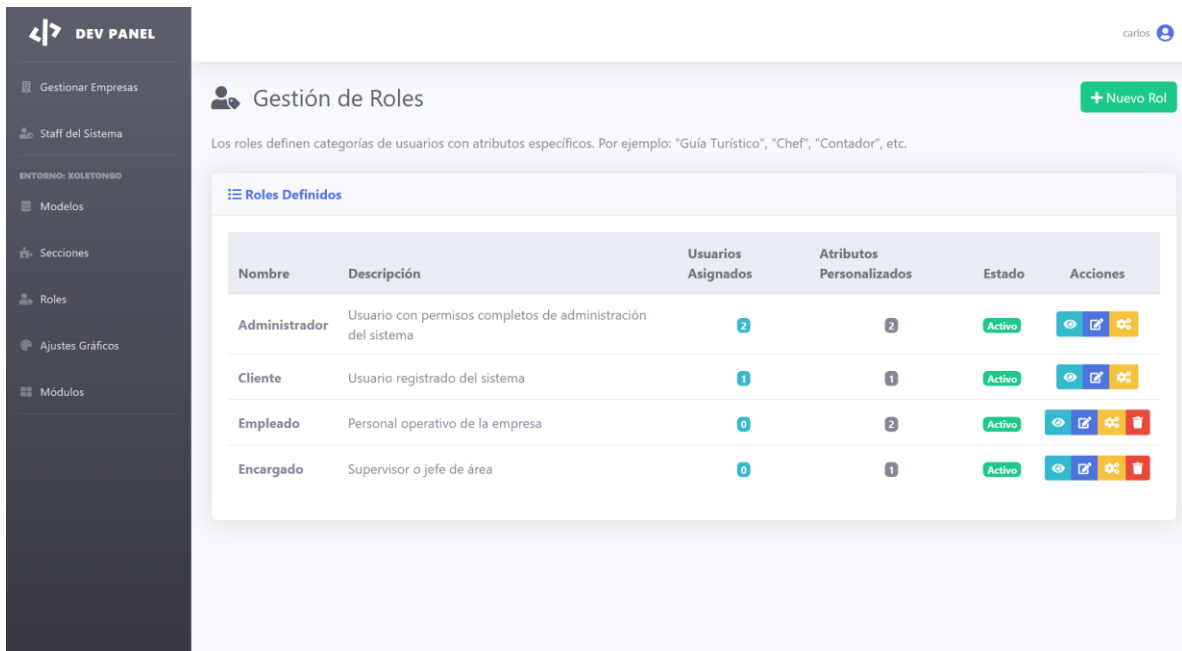


Figura 66 Gestión de roles

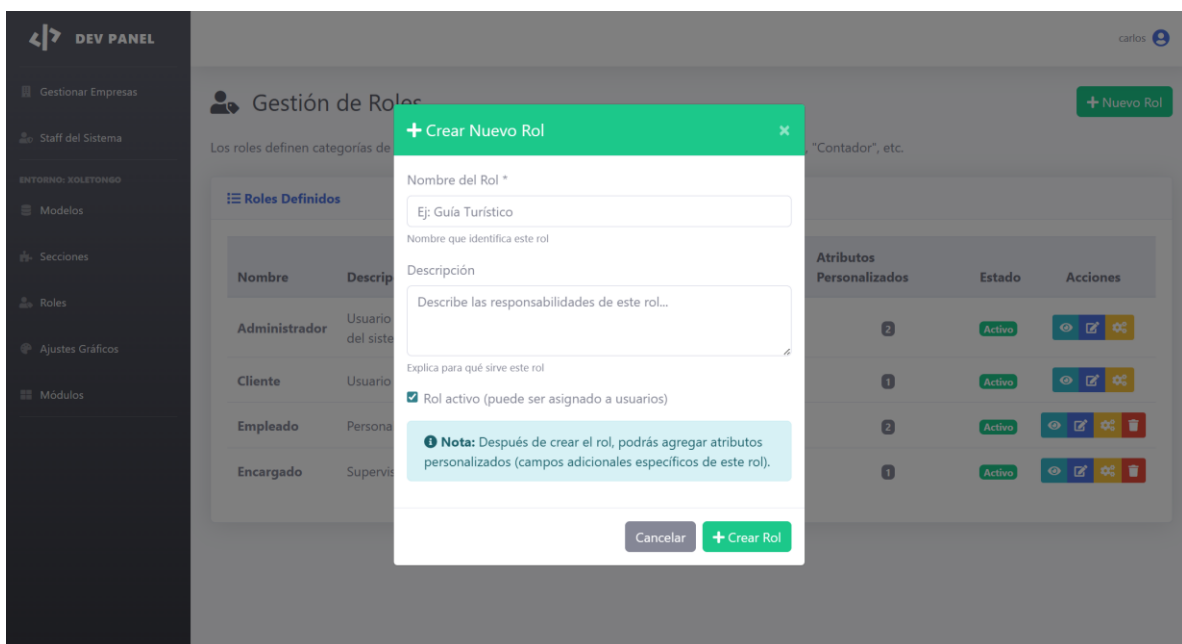


Figura 67 Crear nuevo rol

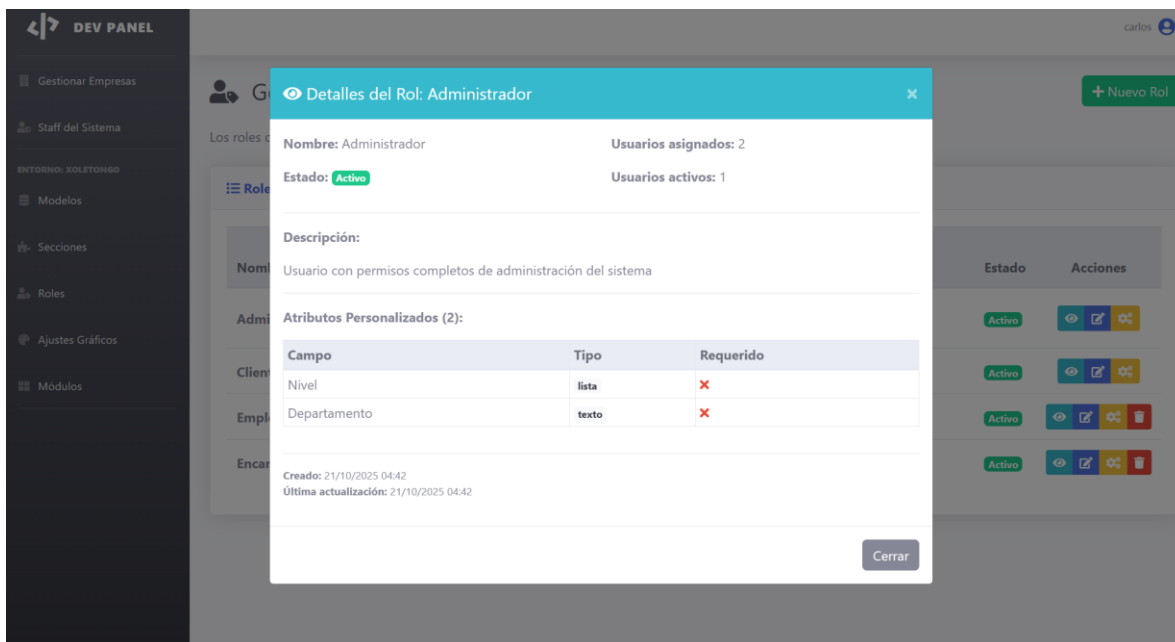


Figura 68 Detalles del rol

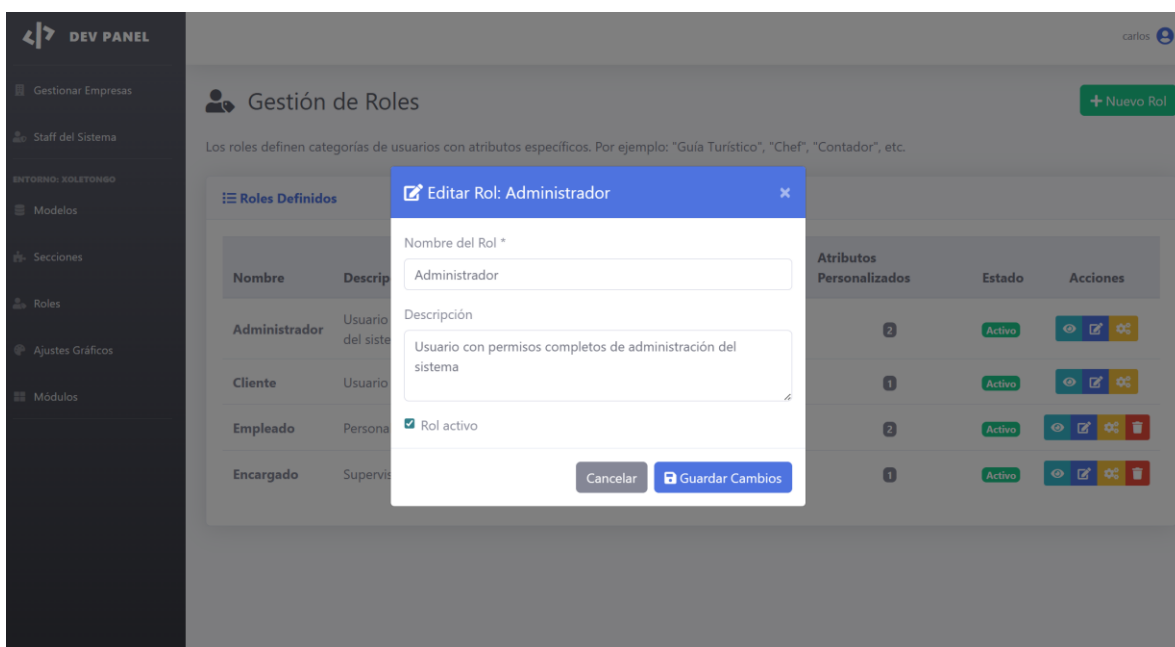


Figura 69 Editar rol

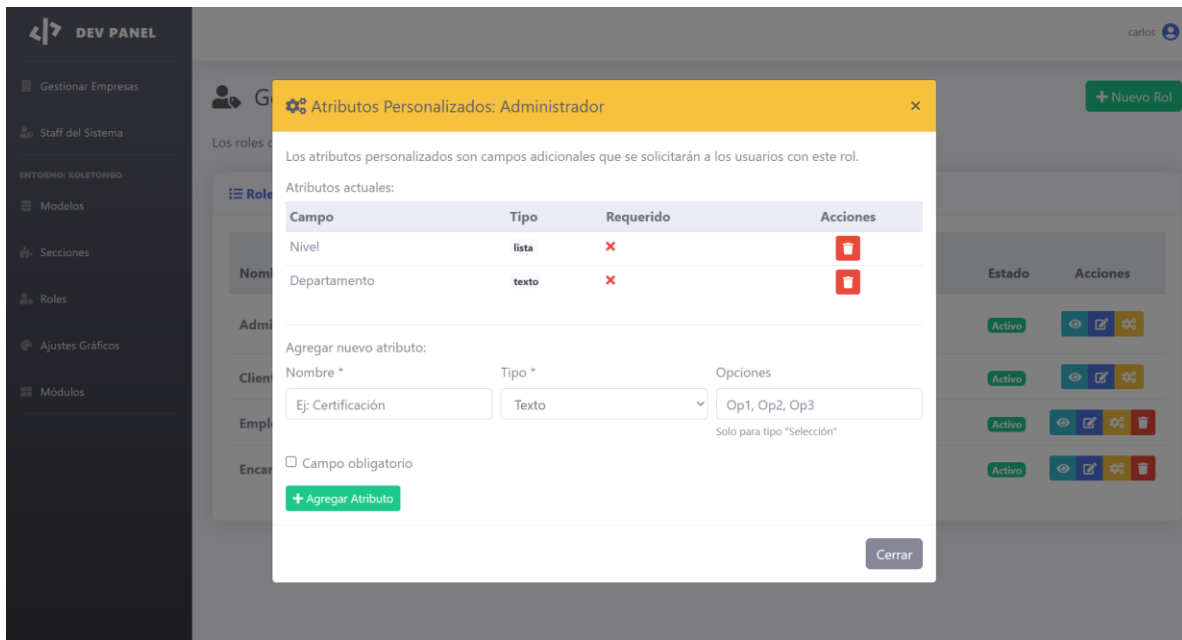


Figura 70 Editar y agregar atributos

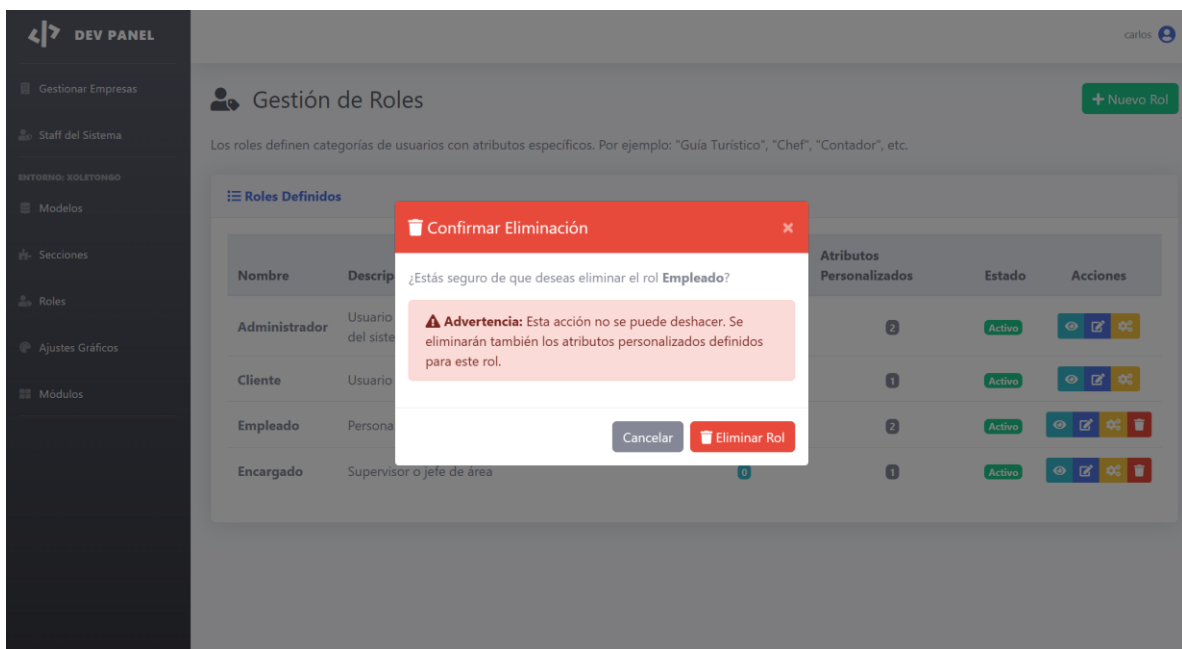


Figura 71 Eliminar rol

Asignar Rol a Usuario

1. Ir a la gestión de usuarios
2. Al crear o editar un usuario, seleccionar el rol deseado
3. Si el rol tiene atributos personalizados, aparecerán campos adicionales

4. Completar los campos requeridos según el schema del rol
5. Guardar el usuario

Atributos Personalizados para Productos y Servicios

Crear Atributo Personalizado

1. Hacer clic en "Atributos" → "Productos" o "Servicios"
2. Hacer clic en "Crear Nuevo Atributo"
3. **Completar formulario:**
 - a. Nombre del atributo (ej: "Talla", "Color", "Material")
 - b. Tipo de dato (texto, número, fecha, boolean)
 - c. Descripción (opcional)
 - d. Aplicar a: Productos o Servicios
4. Hacer clic en "Guardar"
5. El atributo aparecerá automáticamente en los formularios de productos/servicios

Usar Atributos en Productos/Servicios

1. Al crear o editar un producto o servicio, los atributos personalizados aparecerán automáticamente
2. Completar los valores según corresponda
3. Los atributos se guardarán como JSON y se mostrarán en el frontend

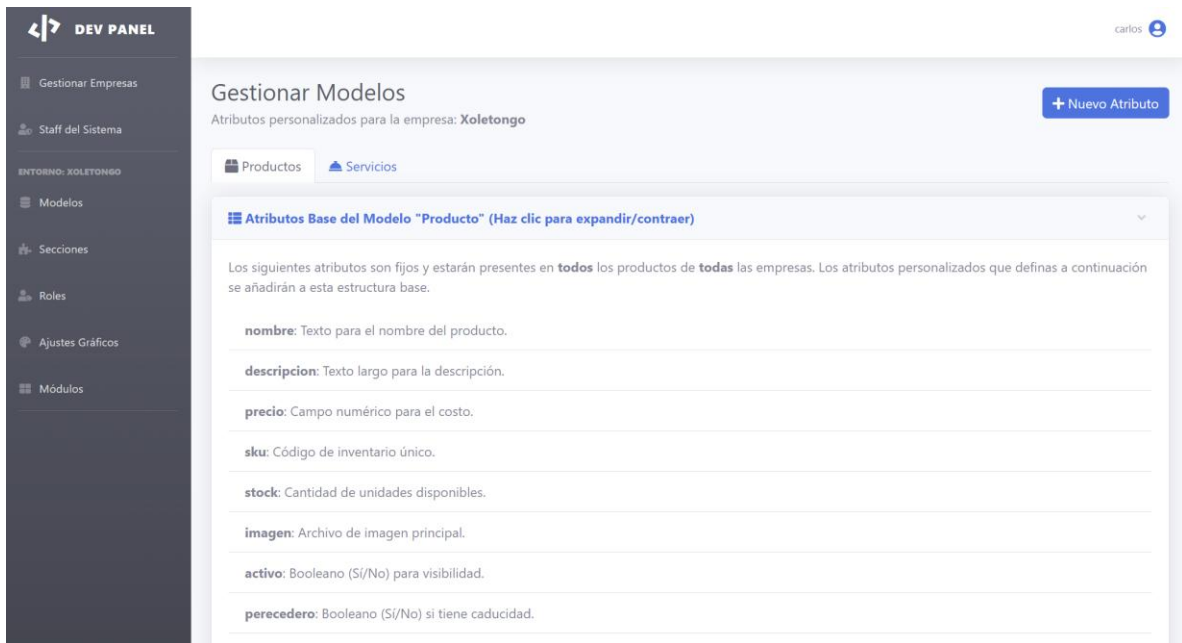


Figura 72 Gestionar módulos

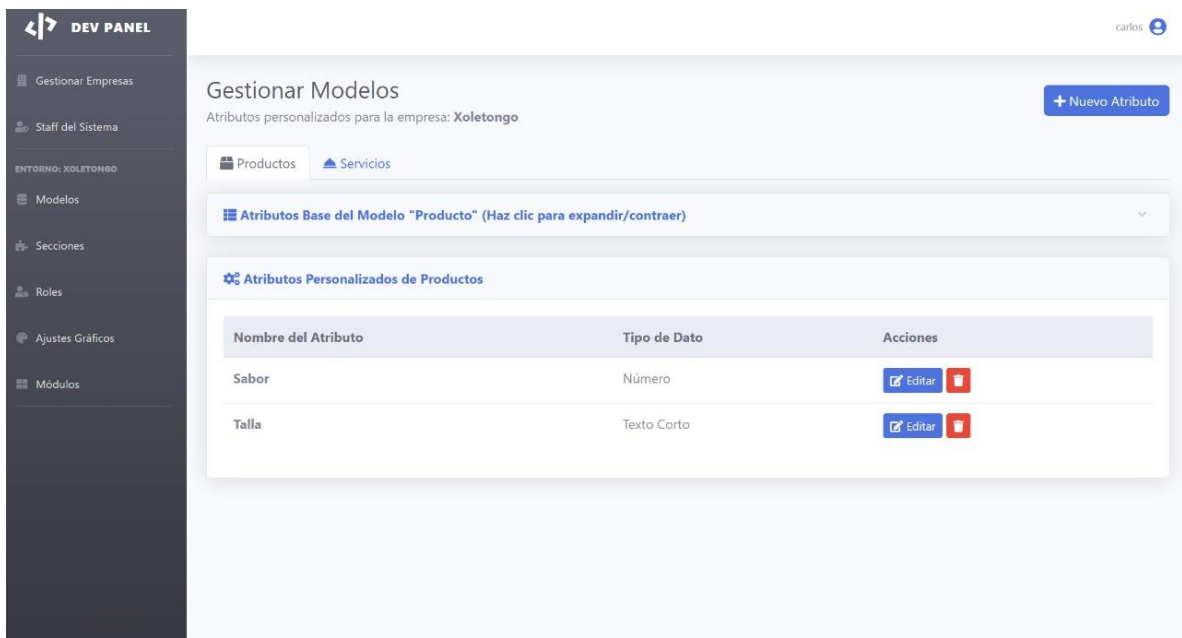
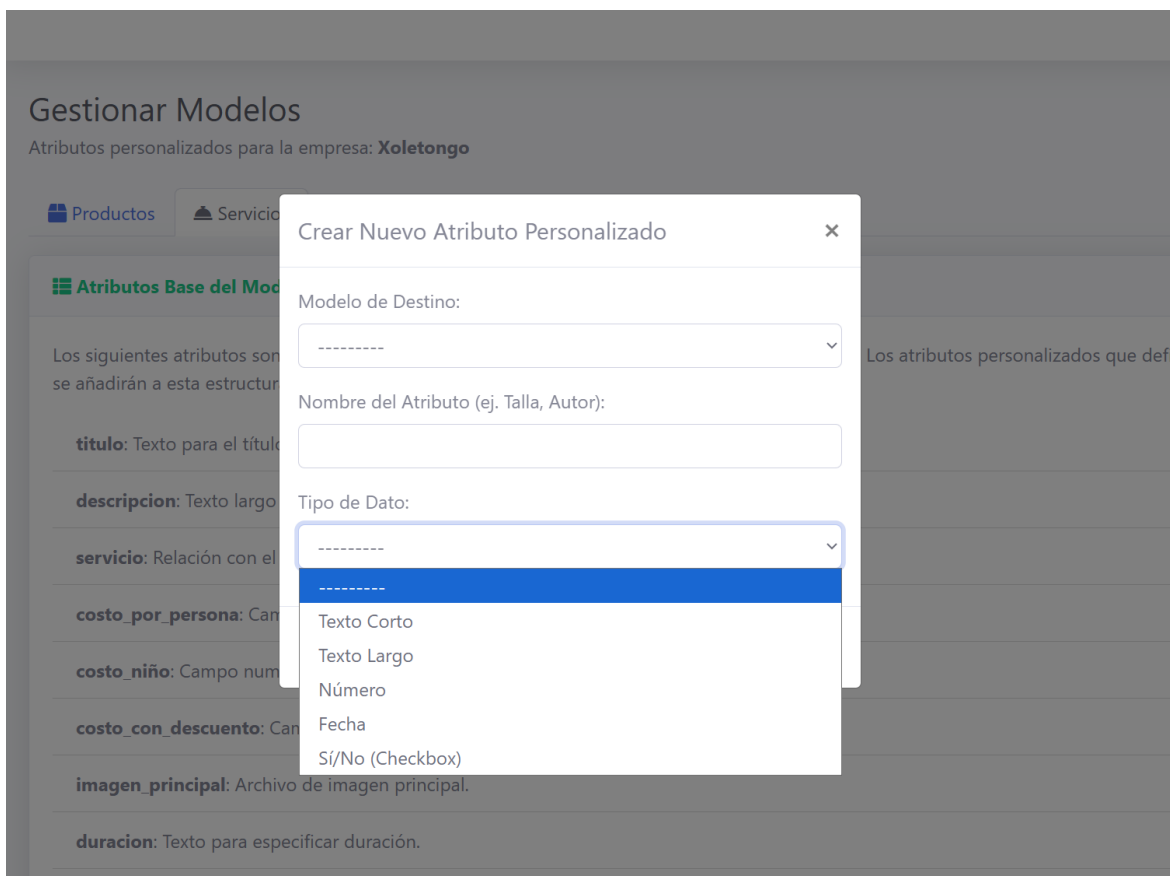


Figura 73 Atributos personalizados en la gestión de los modelos



Atributos Personalizados de Servicios		
Nombre del Atributo	Tipo de Dato	Acciones
Dias	Número	Editar Eliminar
Dificultad	Número	Editar Eliminar
Incluye Comida:	Si/No (Checkbox)	Editar Eliminar

Figura 74 Atributos personalizados de servicios



Gestionar Modelos
Atributos personalizados para la empresa: **Xoletongo**

Crear Nuevo Atributo Personalizado

Modelo de Destino:

Nombre del Atributo (ej. Talla, Autor):

Tipo de Dato:

-
-
- Texto Corto
- Texto Largo
- Número
- Fecha
- Si/No (Checkbox)

Atributos Base del Modelo

Los siguientes atributos son los que se añadirán a esta estructura:

- título:** Texto para el título
- descripcion:** Texto largo
- servicio:** Relación con el servicio
- costo_por_persona:** Campo numérico
- costo_niño:** Campo numérico
- costo_con_descuento:** Campo numérico
- imagen_principal:** Archivo de imagen principal.
- duracion:** Texto para especificar duración.

Los atributos personalizados que definen el modelo son:

Figura 75 Agregar nuevo atributo personalizado

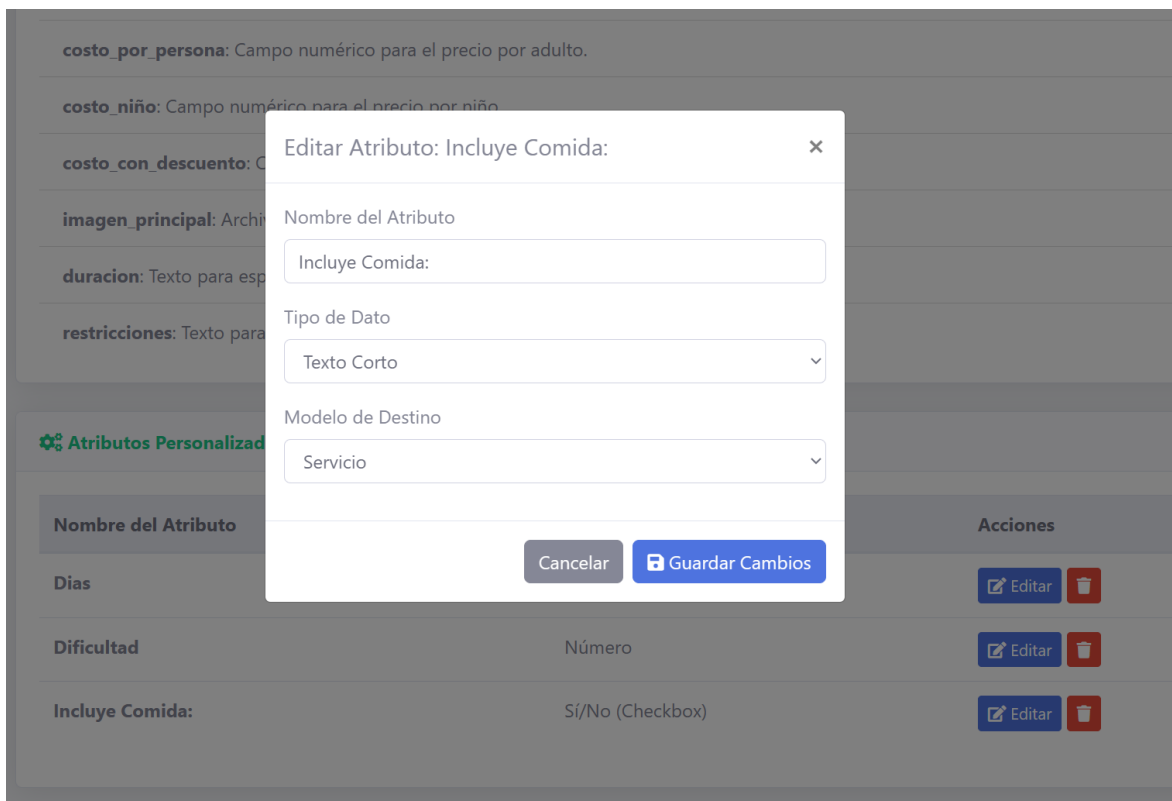


Figura 76 Agregar nuevo atributo personalizado

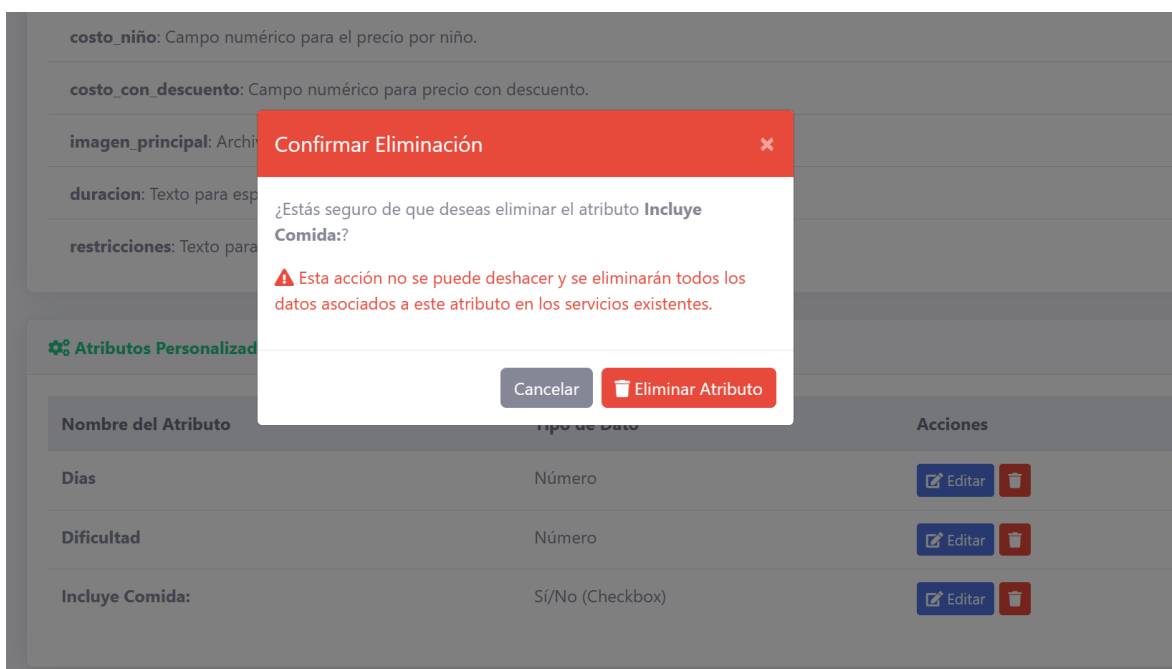


Figura 77 Eliminar atributo

Comandos de Gestión (Management Commands)

Los comandos se ejecutan desde la terminal en el directorio del proyecto:

Cargar Contenido Inicial

```
# Contenido de la página Home
python manage.py seed_xoletongo_content
```

```
# Contenido de la página Nosotros
python manage.py seed_nosotros_content
```

```
# Contenido de la página Servicios
python manage.py seed_servicios_content
```

```
# Contenido de la página Contacto
python manage.py seed_contacto_content
```

```
# Contenido de la página Galería
python manage.py seed_galeria_content
```

```
# Crear página Footer
python manage.py crear_pagina_footer
```

```
# Crear página Productos
python manage.py crear_pagina_productos
```

Comandos de Utilidad

```
# Migrar roles antiguos al nuevo sistema
python manage.py migrar_rols_legacy
```

```
# Actualizar tema de la empresa
python manage.py actualizar_tema_xoletongo
```

```
# Personalizar botón de contacto
python manage.py actualizar_color_boton_contacto
```

Gestión de Superusuarios

1. Hacer clic en "Staff" → "Gestionar Superusuarios"

2. **Crear superusuario:**

- a. Hacer clic en "Crear Superusuario"
- b. Completar formulario (nombre de usuario, email, contraseña)
- c. Guardar

3. Editar superusuario:

- a. Seleccionar de la lista
- b. Modificar información
- c. Guardar cambios



Figura 78 Activar y desactivar staff del sistema

The image shows a web application interface with a sidebar menu on the left containing items like 'Gestionar Empresas', 'Staff del Sistema', 'Modelos', 'Secciones', 'Roles', 'Ajustes Gráficos', and 'Módulos'. The main content area displays a modal titled 'Crear Nuevo Superusuario'. The modal has a green header bar with a close button. It contains several input fields: 'Nombre de usuario *' (filled with 'admin_usuario'), 'Email *' (filled with 'admin@ejemplo.com'), 'Nombre' (filled with 'Juan'), 'Apellidos' (filled with 'Pérez'), 'Contraseña *', and 'Confirmar contraseña *'. Below these fields, there is a 'Requisitos' section with three bullet points: 'Mínimo 8 caracteres', 'Debe incluir letras y números', and 'No puede ser muy común (ej: "12345678")'. There is also a checkbox labeled 'Usuario activo' which is checked. At the bottom of the modal, there is a yellow warning box that says 'Atención: Los superusuarios tienen acceso completo al sistema, incluyendo todas las empresas y configuraciones.' and two buttons: 'Cancelar' and 'Crear Superusuario'.

DEV PANEL

Staff del Sistema

Los superusuarios tienen acceso completo al sistema.

usuario1

Nombre: usuario1

Email: user@ejemplo.com

Creado: 21/10/2025

Editar Activar

Crear Nuevo Superusuario

Nombre de usuario *

admin_usuario

Email *

admin@ejemplo.com

Nombre

Juan

Apellidos

Pérez

Contraseña *

Confirmar contraseña *

Requisitos:

- Mínimo 8 caracteres
- Debe incluir letras y números
- No puede ser muy común (ej: "12345678")

☒ Usuario activo

Atención: Los superusuarios tienen acceso completo al sistema, incluyendo todas las empresas y configuraciones.

Cancelar Crear Superusuario

Figura 79 Crear nuevo superusuario

The image shows the same web application interface as Figure 79, but with a modal titled 'Editar: usuario1' open. The modal has a blue header bar with a close button. It contains input fields for 'Nombre de usuario *' (filled with 'usuario1'), 'Email *' (filled with 'user@ejemplo.com'), 'Nombre' (filled with 'usuario1'), and 'Apellidos' (empty). There is also a checkbox labeled 'Usuario activo' which is unchecked. At the bottom of the modal, there is a light blue note box that says 'Nota: Para cambiar la contraseña, contacta al administrador del sistema.' and two buttons: 'Cancelar' and 'Guardar Cambios'.

Staff del Sistema

Los superusuarios tienen acceso completo al sistema.

usuario1

Nombre: usuario1

Email: user@ejemplo.com

Creado: 21/10/2025

Editar Activar

Editar: usuario1

Nombre de usuario *

usuario1

Email *

user@ejemplo.com

Nombre

usuario1

Apellidos

☐ Usuario activo

Nota: Para cambiar la contraseña, contacta al administrador del sistema.

Cancelar Guardar Cambios

Figura 80 Editar usuario

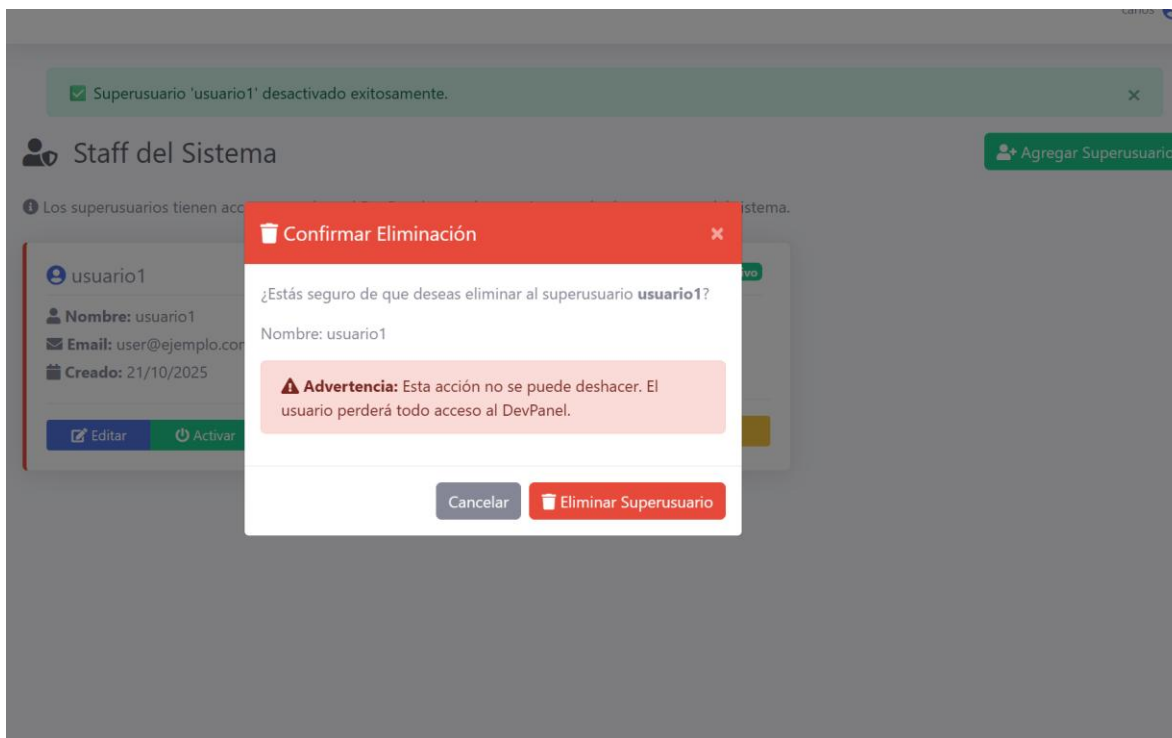


Figura 81 Eliminar usuario

Anexo 2. Manual Técnico

1. Introducción

1.1 Descripción del Sistema

El Sistema Integral de Gestión Operativa Web es una plataforma web desarrollada en Django 5.2 diseñada para la gestión integral de empresas del sector turístico. El sistema proporciona funcionalidades completas de e-commerce, gestión de servicios, reservaciones, y administración de contenido dinámico.

1.2 Características Principales

- **Arquitectura Multi-Tenant:** Soporte para múltiples empresas en una sola instalación
- **E-commerce Completo:** Catálogo de productos, carrito de compras, checkout y gestión de pedidos

- **Sistema de Reservaciones:** Gestión de servicios turísticos con reservaciones en línea
- **Contenido Dinámico:** Páginas y secciones completamente configurables sin modificar código
- **Temas Personalizables:** Sistema de temas con colores, tipografías y estilos configurables
- **Roles Dinámicos:** Sistema de roles personalizados con atributos configurables
- **Panel de Administración:** Dashboard con métricas y gestión completa de operaciones
- **Panel de Desarrollo:** Herramientas avanzadas para personalización y configuración

1.3 Tecnologías Utilizadas

- **Backend:** Django 5.2 (Python 3.13+)
- **Base de Datos:** MySQL 5.7+ / MariaDB 10.3+
- **Frontend:** HTML5, CSS3, JavaScript, Bootstrap
- **Librerías Principales:**
 - PyMySQL 1.1.1 (conexión a MySQL)
 - Pillow 11.2.1 (procesamiento de imágenes)
 - ReportLab 4.4.2 (generación de PDFs)
 - django-widget-tweaks 1.5.0 (mejoras en formularios)
 - paypalrestsdk 1.13.3 (integración PayPal)

2. Arquitectura del Sistema

2.1 Arquitectura General

El sistema sigue una arquitectura **MVC (Modelo-Vista-Controlador)** implementada por Django:

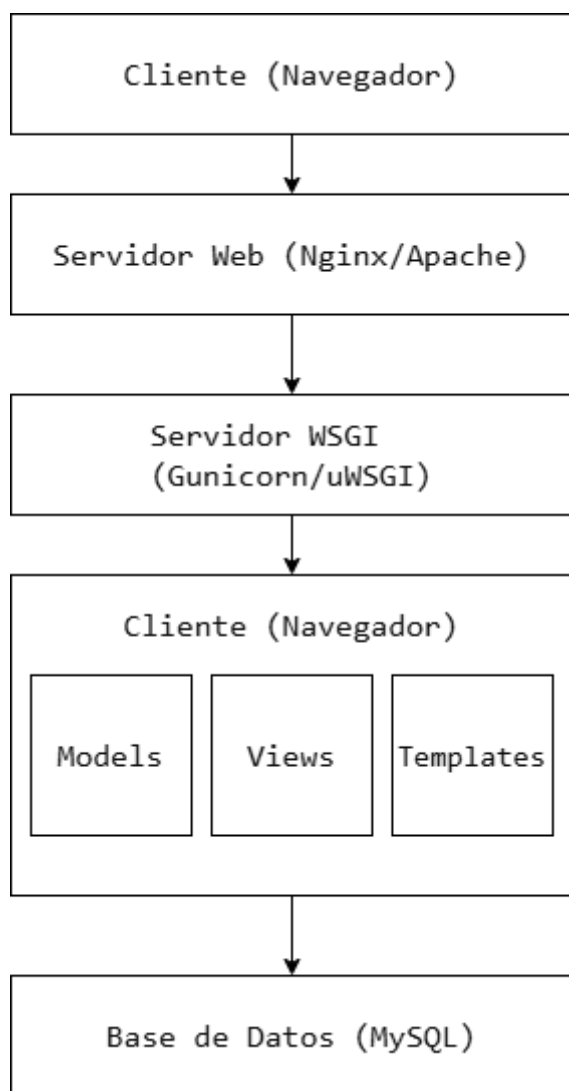


Figura 82 Arquitectura general

2.2 Arquitectura Multi-Tenant

El sistema implementa un modelo **multi-tenant** donde:

- Cada empresa tiene su propio conjunto de datos aislados

- Los modelos principales tienen una relación ForeignKey con Empresa
- Solo una empresa puede estar "activa" a la vez en el sitio público
- Cada empresa puede tener su propio tema, páginas y configuración

2.3 Módulos del Sistema

El sistema está organizado en las siguientes aplicaciones Django:

1. **home**: Páginas públicas, e-commerce, carrito de compras
2. **adminpanel**: Panel de administración para gestión operativa
3. **devpanel**: Panel de desarrollo para personalización avanzada
4. **empresas**: Gestión de empresas y temas
5. **servicios**: Catálogo y gestión de servicios turísticos
6. **reservaciones**: Sistema de reservaciones
7. **usuarios**: Autenticación y gestión de usuarios

2.4 Flujo de Datos

Usuario → URL → View → Model → Base de Datos

↓

Template → HTML → Usuario

3. Requisitos del Sistema

3.1 Requisitos de Hardware

Desarrollo

- **Procesador:** Intel Core i3 o equivalente (2 GHz)
- **Memoria RAM:** 4 GB mínimo, 8 GB recomendado
- **Almacenamiento:** 10 GB de espacio libre
- **Conexión a Internet:** Para descargar dependencias

Producción

- **Procesador:** Intel Core i5 o superior, 4 núcleos
- **Memoria RAM:** 8 GB mínimo, 16 GB recomendado
- **Almacenamiento:** 50 GB SSD
- **Conexión a Internet:** Banda ancha estable

3.2 Requisitos de Software

Sistema Operativo

- Windows 10 o superior
- Linux: Ubuntu 20.04 LTS o superior, Debian 11 o superior
- macOS: macOS 10.15 (Catalina) o superior

Software Base

- **Python:** 3.13 o superior
- **MySQL:** 5.7 o superior (o MariaDB 10.3+)
- **pip:** Gestor de paquetes de Python (incluido con Python)
- **virtualenv:** Opcional pero recomendado
- **Git:** Para control de versiones

Servidor Web (Producción)

- **Nginx** o **Apache** como servidor web reverso
- **Gunicorn** o **uWSGI** como servidor WSGI
- **Systemd** o **Supervisor** para gestión de procesos

4. Instalación y Configuración

4.1 Preparación del Entorno

Clonar o Descargar el Proyecto

Si el proyecto está en un repositorio Git:

```
git clone [URL_DEL_REPOSITORIO]
cd Xoletongo
```

Crear Entorno Virtual

Windows:

```
python -m venv mienv
mienv\Scripts\activate
```

Linux/macOS:

```
python3 -m venv mienv
source mienv/bin/activate
```

4.2 Instalación de Dependencias

8. Navegar al directorio del proyecto:

```
cd xoletongo
```

9. Instalar dependencias:

```
pip install -r requirements.txt
```

Nota: Si requirements.txt no existe, instalar manualmente:

```
pip install Django==5.2
pip install PyMySQL==1.1.1
pip install Pillow==11.2.1
pip install reportlab==4.4.2
```

```
pip install django-widget-tweaks==1.5.0
pip install cryptography
```

4.3 Configuración de Base de Datos

Crear Base de Datos MySQL

10. Acceder a MySQL:

```
mysql -u root -p
```

11. Crear base de datos:

```
CREATE DATABASE xoletongo CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
```

12. Crear usuario (opcional):

```
CREATE USER 'userD'@'localhost' IDENTIFIED BY '123456';
GRANT ALL PRIVILEGES ON xoletongo.* TO 'userD'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

Configurar Django para MySQL

13. Abrir xoletongo/xoleongo/settings.py

14. Verificar configuración de base de datos:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'xoletongo',
        'USER': 'userD',
        'PASSWORD': '123456',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {
```

```

        'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        'charset': 'utf8mb4',
    },
}
}

```

15. Si se usa PyMySQL, agregar al inicio de `xoletongo/xoleongo/__init__.py`:

```

import pymysql
pymysql.install_as_MySQLdb()

```

4.4 Ejecutar Migraciones

```
python manage.py migrate
```

Este comando creará todas las tablas necesarias en la base de datos.

4.5 Crear Superusuario

```
python manage.py createsuperuser
```

Seguir las instrucciones para crear el usuario administrador.

4.6 Crear Empresa Inicial

El sistema requiere al menos una empresa activa. Crear desde la shell de Django:

```
python manage.py shell
```

```

from empresas.models import Empresa
empresa = Empresa.objects.create(
    nombre='Tu Empresa',
    nombre_titular='Nombre del Titular',

```

```

        correo_contacto='contacto@tuempresa.com',
        telefono='1234567890',
        ubicacion='Dirección de la empresa',
        cuenta_bancaria='123456789012345678',
        clabe='123456789012345678',
        numero_terjeta='1234567890123456',
        activa=True # IMPORTANTE: marcar como activa
    )
    print(f"Empresa '{empresa.nombre}' creada exitosamente")
    exit()

```

4.7 Cargar Contenido Inicial (Opcional)

```

# Contenido de la página Home
python manage.py seed_xoletongo_content

# Página Nosotros
python manage.py seed_nosotros_content

# Página Servicios
python manage.py seed_servicios_content

# Página Contacto
python manage.py seed_contacto_content

# Página Galería
python manage.py seed_galeria_content

# Crear página Footer
python manage.py crear_pagina_footer

# Crear página Productos
python manage.py crear_pagina_productos

```

4.8 Recopilar Archivos Estáticos

```
python manage.py collectstatic --noinput
```

4.9 Ejecutar Servidor de Desarrollo

```
python manage.py runserver
```

El servidor estará disponible en:

- **Sitio público:** <http://127.0.0.1:8000/>
- **Panel de administración Django:** <http://127.0.0.1:8000/admin/>
- **Panel de administración:** <http://127.0.0.1:8000/adminpanel/>
- **Panel de desarrollo:** <http://127.0.0.1:8000/dev/>

ADVERTENCIA: El servidor de desarrollo NO es adecuado para producción.

5. Estructura del Proyecto

5.1 Estructura de Directorios

```
xoletongo/
├── xoleongo/          # Configuración principal del proyecto
│   ├── __init__.py
│   ├── settings.py    # Configuración del proyecto
│   ├── urls.py        # URLs principales
│   ├── wsgi.py        # Configuración WSGI
│   └── asgi.py        # Configuración ASGI
│
├── home/              # Aplicación principal (páginas públicas)
│   ├── models.py
│   ├── views.py
│   ├── urls.py
│   ├── templates/
│   └── static/
│
├── adminpanel/        # Panel de administración
│   ├── models.py
│   ├── views.py
│   ├── forms.py
│   ├── urls.py
│   ├── middleware.py
│   ├── templates/
│   └── static/
│
├── devpanel/          # Panel de desarrollo
│   ├── models.py
│   └── views.py
```

```

|   |— forms.py
|   |— urls.py
|   |— management/
|   |   |— commands/    # Comandos personalizados
|   |   |— templates/
|
|— empresas/            # Gestión de empresas y temas
|   |— models.py
|   |— views.py
|   |— backends.py     # Backend SMTP personalizado
|   |— signals.py
|
|— servicios/          # Gestión de servicios turísticos
|   |— models.py
|   |— views.py
|   |— forms.py
|   |— templates/
|
|— reservaciones/      # Sistema de reservaciones
|   |— models.py
|   |— views.py
|   |— forms.py
|   |— templates/
|
|— usuarios/           # Autenticación y usuarios
|   |— models.py
|   |— views.py
|   |— forms.py
|   |— templates/
|
|— templates/          # Templates base

```

```

|   |— base.html
|   |— base_template.html
|
|— static/          # Archivos estáticos (CSS, JS, imágenes)
|   |— css/
|   |— js/
|   |— img/
|
|— media/           # Archivos subidos por usuarios
|   |— productos/
|   |— servicios/
|
|— manage.py        # Script de gestión de Django
|— requirements.txt # Dependencias del proyecto
|— README.md

```

5.2 Archivos de Configuración Principales

settings.py

Archivo principal de configuración ubicado en `xoletongo/xoleongo/settings.py`.
 Contiene:

- Configuración de base de datos
- Aplicaciones instaladas (INSTALLED_APPS)
- Middleware
- Configuración de archivos estáticos y media
- Configuración de seguridad
- Variables de entorno

urls.py

Archivo principal de URLs en xoletongo/xoleongo/urls.py que incluye las URLs de todas las aplicaciones:

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path("", include('home.urls')),  
    path("", include('servicios.urls')),  
    path('usuarios/', include('usuarios.urls')),  
    path("", include('reservaciones.urls')),  
    path('adminpanel/', include('adminpanel.urls')),  
    path('dev/', include('devpanel.urls')),  
]
```

6. Configuración de Base de Datos

6.1 Modelos Principales

Empresa

- **Tabla:** empresas_empresa
- **Relaciones:** Tiene muchos usuarios, productos, servicios, pedidos, reservaciones
- **Campos clave:** nombre, activa, tema, productos_habilitado, servicios_habilitado

CustomUser

- **Tabla:** usuarios_customuser
- **Herencia:** AbstractUser de Django

- **Relaciones:** Pertenece a una Empresa, tiene un Rol
- **Campos clave:** username, email, empresa, rol, atributos_personalizados

Producto

- **Tabla:** adminpanel_producto
- **Relaciones:** Pertenece a una Empresa y una CategoriaProducto
- **Campos clave:** nombre, precio, stock, sku, atributos_personalizados

Pedido

- **Tabla:** adminpanel_pedido
- **Relaciones:** Pertenece a una Empresa y un CustomUser (opcional)
- **Campos clave:** numero_pedido, total, estado, metodo_pago

Servicio

- **Tabla:** servicios_servicio
- **Relaciones:** Pertenece a una Empresa y un TipoServicio
- **Campos clave:** titulo, costo_por_persona, costo_niño, costo_con_descuento

Reservacion

- **Tabla:** reservaciones_reservacion
- **Relaciones:** Pertenece a una Empresa, tiene muchos Servicio (Many-to-Many)
- **Campos clave:** nombre_cliente, fecha_inicio, fecha_fin, estado

6.2 Relaciones Principales

Empresa (1) —→ (N) CustomUser

Empresa (1) —→ (N) Producto

Empresa (1) —→ (N) Servicio

Empresa (1) —→ (N) Pedido

Empresa (1) —→ (N) Reservacion
Empresa (1) —→ (1) Tema
CustomUser (N) —→ (1) Empresa
CustomUser (N) —→ (1) Rol
CustomUser (1) —→ (N) Pedido
Producto (N) —→ (1) CategoriaProducto
Pedido (1) —→ (N) ItemPedido
ItemPedido (N) —→ (1) Producto
Reservacion (N) —→ (M) Servicio (a través de Reservacion_servicio)

6.3 Índices y Optimización

Los índices principales están definidos en los modelos Django. Para optimización adicional:

-- Ejemplo de índices adicionales

```
CREATE INDEX idx_producto_empresa ON adminpanel_producto(empresa_id);  
CREATE INDEX idx_pedido_estado ON adminpanel_pedido(estado);  
CREATE INDEX idx_reservacion_fecha ON  
reservaciones_reservacion(fecha_inicio);
```

7. Módulos Principales

7.1 Módulo Home

Ubicación: xoletongo/home/

Funcionalidades:

- Página de inicio con secciones dinámicas
- Catálogo de productos
- Detalle de productos
- Carrito de compras
- Proceso de checkout
- Formulario de contacto
- Páginas estáticas (Nosotros, Galería)

Vistas principales:

- index(): Página principal
- productos(): Catálogo de productos
- producto_detalle(): Detalle de producto
- agregar_al_carrito(): Agregar producto al carrito
- ver_carrito(): Ver carrito de compras
- checkout(): Proceso de pago
- procesar_pedido(): Procesar pedido final

7.2 Módulo AdminPanel

Ubicación: xoletongo/adminpanel/

Funcionalidades:

- Dashboard con métricas

- Gestión de usuarios (CRUD)
- Gestión de productos (CRUD)
- Gestión de pedidos
- Gestión de servicios
- Gestión de categorías
- Exportación a PDF

Vistas principales:

- dashboard(): Dashboard principal
- lista_usuarios(): Lista de usuarios
- admin_productos(): Lista de productos
- admin_agregar_producto(): Crear producto
- editar_producto(): Editar producto
- exportar_productos_pdf(): Exportar productos a PDF

7.3 Módulo DevPanel

Ubicación: xoletongo/devpanel/

Funcionalidades:

- Gestión de empresas
- Editor de temas
- Gestión de páginas y secciones
- Gestión de roles dinámicos
- Gestión de atributos personalizados

Vistas principales:

- gestion_empresas(): Lista de empresas
- crear_empresa(): Crear empresa
- editar_empresa(): Editar empresa
- gestion_secciones(): Gestión de secciones

- crear_seccion(): Crear sección
- gestion_rol(): Gestión de roles
- crear_rol(): Crear rol personalizado

7.4 Módulo Empresas

Ubicación: xoletongo/empresas/

Funcionalidades:

- Modelo de empresa
- Modelo de tema
- Backend SMTP personalizado por empresa
- Señales para inicialización

Modelos principales:

- Empresa: Información de la empresa
- Tema: Configuración visual del tema

7.5 Módulo Servicios

Ubicación: xoletongo/servicios/

Funcionalidades:

- Catálogo de servicios turísticos
- Tipos de servicios (avistamiento, hospedaje, tours, etc.)
- Gestión de servicios desde admin

Modelos principales:

- TipoServicio: Categorías de servicios
- Servicio: Servicios individuales

7.6 Módulo Reservasiones

Ubicación: xoletongo/reservaciones/

Funcionalidades:

- Sistema de reservaciones
- Gestión de reservaciones
- Cálculo automático de costos

Modelos principales:

- Reservacion: Reservación de servicios
- Reservacion_servicio: Relación Many-to-Many entre reservaciones y servicios

7.7 Módulo Usuarios

Ubicación: xoletongo/usuarios/

Funcionalidades:

- Autenticación de usuarios
- Registro de usuarios
- Perfil de usuario
- Sistema de roles

Vistas principales:

- registro_view(): Registro de usuarios
- login_view(): Inicio de sesión
- logout_view(): Cerrar sesión
- perfil_view(): Perfil de usuario

8. APIs y Endpoints

8.1 URLs Principales

Públicas (Home)

- / - Página principal
- /nosotros/ - Página nosotros
- /productos/ - Catálogo de productos
- /producto/<id>/ - Detalle de producto
- /servicios/ - Catálogo de servicios
- /galeria/ - Galería de imágenes
- /contacto/ - Formulario de contacto

Autenticación (Usuarios)

- /usuarios/registro/ - Registro
- /usuarios/login/ - Inicio de sesión
- /usuarios/logout/ - Cerrar sesión
- /usuarios/perfil/ - Perfil de usuario

Panel de Administración

- /adminpanel/ - Dashboard
- /adminpanel/usuarios/ - Gestión de usuarios
- /adminpanel/productos/ - Gestión de productos
- /adminpanel/pedidos/ - Gestión de pedidos
- /adminpanel/servicios/ - Gestión de servicios

Panel de Desarrollo

- /dev/empresas/ - Gestión de empresas
- /dev/secciones/ - Gestión de secciones
- /dev/roles/ - Gestión de roles
- /dev/modelos/ - Gestión de atributos personalizados

8.2 Endpoints de E-commerce

Carrito de Compras

- POST /agregar-al-carrito/<producto_id>/ - Agregar producto al carrito
- GET /carrito/ - Ver carrito
- POST /carrito/actualizar/ - Actualizar cantidades
- POST /carrito/eliminar/<item_id>/ - Eliminar item

Checkout

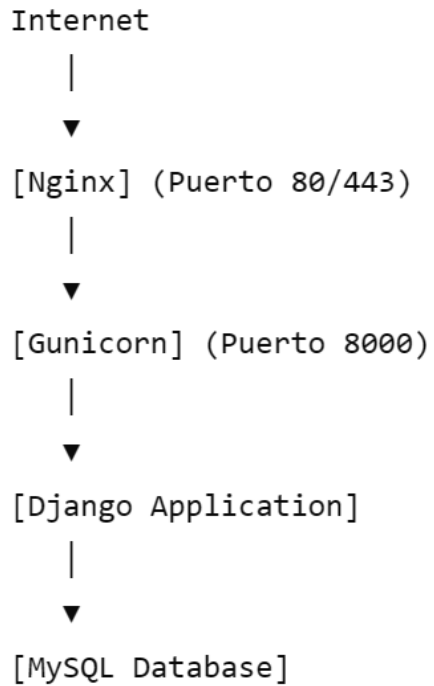
- GET /checkout/ - Página de checkout
- POST /procesar-pedido/ - Procesar pedido

8.3 Endpoints de Reservaciones

- GET /reservaciones/ - Lista de reservaciones
- POST /reservaciones/crear/ - Crear reservación
- GET /reservaciones/<id>/ - Detalle de reservación
- POST /reservaciones/<id>/actualizar/ - Actualizar reservación

9. Despliegue en Producción

9.1 Arquitectura Recomendada



9.2 Configuración con Nginx y Gunicorn

1. Instalar Nginx

```
sudo apt update  
sudo apt install nginx
```

2. Instalar Gunicorn

```
pip install gunicorn
```

3. Configurar Gunicorn

Crear archivo `gunicorn_config.py` en el directorio del proyecto:

```
bind = "127.0.0.1:8000"  
workers = 4  
worker_class = "sync"
```

timeout = 120

keepalive = 5

4. Crear Servicio Systemd

Crear /etc/systemd/system/xoletongo.service:

[Unit]

Description=Gunicorn daemon for Xoletongo

After=network.target

[Service]

User=www-data

Group=www-data

WorkingDirectory=/ruta/al/proyecto/xoletongo

ExecStart=/ruta/al/venv/bin/gunicorn --config gunicorn_config.py

xoleongo.wsgi:application

[Install]

WantedBy=multi-user.target

Activar servicio:

sudo systemctl daemon-reload

sudo systemctl enable xoletongo

sudo systemctl start xoletongo

5. Configurar Nginx

Crear /etc/nginx/sites-available/xoletongo:

server {

listen 80;

server_name tudominio.com www.tudominio.com;

location / {

```

    proxy_pass http://127.0.0.1:8000;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
location /static/ {
    alias /ruta/al/proyecto/xoletongo/staticfiles/;
}
location /media/ {
    alias /ruta/al/proyecto/xoletongo/media/;
}
}

```

Habilitar sitio:

```

sudo ln -s /etc/nginx/sites-available/xoletongo /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx

```

6. Configurar SSL con Let's Encrypt

```

sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d tudominio.com -d www.tudominio.com

```

6.1 Configuración de Seguridad en Producción

Modificar settings.py:

```
# Cambiar SECRET_KEY
SECRET_KEY = 'tu-secret-key-muy-segura-aqui-generar-nueva'
# Desactivar DEBUG
DEBUG = False
# Configurar ALLOWED_HOSTS
ALLOWED_HOSTS = ['tudominio.com', 'www.tudominio.com', 'IP-DEL-SERVIDOR']
# Configuración de seguridad adicional
SECURE_SSL_REDIRECT = True
SESSION_COOKIE_SECURE = True
CSRF_COOKIE_SECURE = True
SECURE_BROWSER_XSS_FILTER = True
SECURE_CONTENT_TYPE_NOSNIFF = True
X_FRAME_OPTIONS = 'DENY'
```

Generar SECRET_KEY seguro:

```
from django.core.management.utils import get_random_secret_key
print(get_random_secret_key())
```

6.2 Configuración de Firewall

```
sudo ufw allow 'Nginx Full'
sudo ufw allow SSH
sudo ufw enable
```

7. Mantenimiento

7.1 RespalDOS de Base de Datos

Backup Manual

```
mysqldump -u userD -p xoletongo > backup_$(date +%Y%m%d_%H%M%S).sql
```

Backup Automático con Cron

Crear script /ruta/al/backup.sh:

```
#!/bin/bash
BACKUP_DIR="/ruta/al/backups"
DATE=$(date +%Y%m%d_%H%M%S)
mysqldump -u userD -p123456 xoletongo > $BACKUP_DIR/backup_$DATE.sql
# Mantener solo los últimos 7 días
find $BACKUP_DIR -name "backup_*.sql" -mtime +7 -delete
```

Hacer ejecutable y agregar a crontab:

```
chmod +x /ruta/al/backup.sh
crontab -e
# Agregar línea: 0 2 * * * /ruta/al/backup.sh
```

7.2 RespalDOS de Archivos Media

```
tar -czf media_backup_$(date +%Y%m%d).tar.gz
/ruta/al/proyecto/xoletongo/media/
```

7.3 Actualización del Sistema

16. Respalidar base de datos y archivos

17. Activar entorno virtual

18. Actualizar código:

git pull origin main # Si se usa Git

19. Actualizar dependencias:

pip install -r requirements.txt --upgrade

20. Ejecutar migraciones:

python manage.py migrate

21. Recopilar archivos estáticos:

python manage.py collectstatic --noinput

22. Reiniciar servicios:

sudo systemctl restart xoletongo

sudo systemctl restart nginx

7.4 Monitoreo de Logs

Configurar Logging en settings.py

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'file': {
            'level': 'INFO',
            'class': 'logging.FileHandler',
            'filename': '/ruta/al/proyecto/logs/django.log',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['file'],
```

```

        'level': 'INFO',
        'propagate': True,
    },
},
}

```

Ver Logs

Logs de Gunicorn

```
sudo journalctl -u xoletongo -f
```

Logs de Nginx

```
sudo tail -f /var/log/nginx/error.log
```

```
sudo tail -f /var/log/nginx/access.log
```

7.5 Optimización de Rendimiento

1. Caché (Recomendado)

Instalar Redis:

```
sudo apt install redis-server
```

```
pip install django-redis
```

Configurar en settings.py:

```

CACHES = {
    'default': {
        'BACKEND': 'django_redis.cache.RedisCache',
        'LOCATION': 'redis://127.0.0.1:6379/1',
        'OPTIONS': {
            'CLIENT_CLASS': 'django_redis.client.DefaultClient',
        }
    }
}

```

2. Optimización de Queries

- Usar `select_related()` y `prefetch_related()` en vistas
- Evitar queries N+1
- Usar `only()` y `defer()` para cargar solo campos necesarios

3. Compresión de Archivos Estáticos

En Nginx, agregar:

```
gzip on;
```

```
gzip_types text/plain text/css application/json application/javascript text/xml  
application/xml;
```

7.6 Limpieza de Datos Temporales

Limpiar Sesiones Expiradas

```
python manage.py clearsessions
```

Agregar a crontab para ejecutar semanalmente:

```
0 3 * * 0 python /ruta/al/proyecto/manage.py clearsessions
```

8. Solución de Problemas

8.1 Errores Comunes

Error: "ModuleNotFoundError: No module named 'pymysql'"

Solución:

```
pip install pymysql
# Agregar al inicio de settings.py o __init__.py:
import pymysql
pymysql.install_as_MySQLdb()
```

Error: "OperationalError: (2003, 'Can't connect to MySQL server')"

Solución:

- Verificar que MySQL esté ejecutándose: `sudo systemctl status mysql`
- Verificar credenciales en settings.py
- Verificar permisos del usuario: `GRANT ALL PRIVILEGES ON xoletongo.* TO 'userD'@'localhost';`

Error: "Permission denied" al subir archivos

Solución:

```
sudo chown -R www-data:www-data /ruta/al/proyecto/media
sudo chmod -R 755 /ruta/al/proyecto/media
```

Error: "Static files not found"

Solución:

```
python manage.py collectstatic
# Verificar STATIC_ROOT y STATIC_URL en settings.py
```

Error: "CSRF verification failed"

Solución:

- Verificar que CSRF_COOKIE_SECURE sea False en desarrollo
- Verificar configuración de ALLOWED_HOSTS
- Limpiar cookies del navegador

Base de Datos Bloqueada

Solución:

```
mysql -u root -p  
SHOW PROCESSLIST;  
KILL [ID_DEL_PROCESO];
```

Problemas con Migraciones

Solución:

```
# Ver estado de migraciones  
python manage.py showmigrations  
# Crear migraciones faltantes  
python manage.py makemigrations  
# Aplicar migraciones  
python manage.py migrate  
# Si hay conflictos, resetear (CUIDADO: pérdida de datos)  
python manage.py migrate --fake-initial
```

8.2 Debugging

Activar Modo Debug

En settings.py:

DEBUG = True

Ver Queries SQL

Agregar en settings.py:

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
        },
    },
    'loggers': {
        'django.db.backends': {
            'handlers': ['console'],
            'level': 'DEBUG',
        },
    },
}
```

9. Seguridad

9.1 Buenas Prácticas de Seguridad

23. Mantener Django y dependencias actualizadas:

```
pip list --outdated
```

```
pip install --upgrade [paquete]
```

24. Cambiar SECRET_KEY en producción

25. Usar HTTPS en producción

26. Configurar firewall apropiadamente

27. Restringir acceso a panel de desarrollo solo a superusuarios

28. Validar y sanitizar todas las entradas del usuario

29. Usar contraseñas seguras para superusuarios

30. Realizar respaldos regulares

31. Monitorear logs para actividades sospechosas

32. Limitar intentos de login (recomendado implementar)

9.2 Configuración de Seguridad en settings.py

```
# Headers de seguridad
```

```
SECURE_CONTENT_TYPE_NOSNIFF = True
```

```
SECURE_BROWSER_XSS_FILTER = True
```

```
X_FRAME_OPTIONS = 'DENY'
```

```
SECURE_HSTS_SECONDS = 31536000
```

```
SECURE_HSTS_INCLUDE_SUBDOMAINS = True
```

```
SECURE_HSTS_PRELOAD = True
```

```
# Cookies seguras (solo en HTTPS)
```

```
SESSION_COOKIE_SECURE = True
```

```
CSRF_COOKIE_SECURE = True
```

```
SESSION_COOKIE_HTTPONLY = True
```

```
CSRF_COOKIE_HTTPONLY = True
```

9.3 Protección de Datos Sensibles

- **Nunca** commitear SECRET_KEY en el repositorio
- Usar variables de entorno para credenciales
- Encriptar datos sensibles en la base de datos
- Implementar rate limiting para APIs

10. Anexos

10.1 Comandos de Gestión Útiles

Crear migraciones

```
python manage.py makemigrations
```

Aplicar migraciones

```
python manage.py migrate
```

Crear superusuario

```
python manage.py createsuperuser
```

Recopilar archivos estáticos

```
python manage.py collectstatic
```

Abrir shell de Django

```
python manage.py shell
```

Verificar configuración

```
python manage.py check
```

Limpiar sesiones expiradas

```
python manage.py clearsessions
```

10.2 Variables de Entorno Recomendadas

Crear archivo .env (no commitear):

```
SECRET_KEY=tu-secret-key-aqui
```

```
DEBUG=False
```

```
ALLOWED_HOSTS=tudominio.com,www.tudominio.com
```

```
DB_NAME=xoletongo
```

DB_USER=userD
DB_PASSWORD=123456
DB_HOST=localhost
DB_PORT=3306

10.3 Estructura de Base de Datos

Para ver la estructura completa de la base de datos:

```
python manage.py sqlmigrate [app] [migration_number]
```

O usar herramientas gráficas como:

- phpMyAdmin
- MySQL Workbench
- DBeaver

10.4 Recursos Adicionales

- **Documentación de Django:** <https://docs.djangoproject.com/>
- **Documentación de MySQL:** <https://dev.mysql.com/doc/>
- **Documentación de Nginx:** <https://nginx.org/en/docs/>
- **Documentación de Gunicorn:** <https://docs.gunicorn.org/>

Conclusión

Este manual técnico proporciona una guía completa para la instalación, configuración, mantenimiento y solución de problemas del Sistema Integral de Gestión Operativa Web. Para consultas adicionales o soporte técnico, consulte la documentación oficial de Django o contacte al equipo de desarrollo.

Versión del Manual: 1.0

Última Actualización: 2025

Sistema: Xoletongo - Sistema Integral de Gestión Operativa Web