PARQUE CIENTÍFICO Y TECNOLÓGICO UTPL

Reto de la ‘Cámara De Comercio Loja’

Versión.1.0.0

Ficha Técnica

Autores:

Líder de grupo	Luis Fernando León Tène
Estudiante	Omer Alexis Benitez Cabrera
Estudiante	Lenin Andrés Cuenca Cuenca

Docente Tutor:

RENE ROLANDO ELIZALDE SOLANO

Índice de contenidos

Índice de figuras

1. Descripción del prototipo

El sistema web desarrollado para la Cámara de Comercio de Loja es un prototipo integral de gestión empresarial que representa un avance significativo en la digitalización de sus procesos administrativos y operativos. Este prototipo tiene como propósito principal modernizar las operaciones tradicionales mediante una plataforma que automatiza la afiliación de empresas y personas naturales, facilita la reserva de servicios institucionales, optimiza la administración de convenios comerciales y mejora la comunicación con los socios a través de notificaciones automáticas y un chatbot interactivo.

Diseñado bajo una metodología ágil de desarrollo, el sistema fue implementado con un enfoque iterativo e incremental que permitió ajustar funcionalidades durante todo el ciclo de creación. Se adoptó la arquitectura Modelo-Vista-Template (MVT) de Django, aprovechando sus ventajas para estructurar de manera ordenada el código, facilitar la mantenibilidad y garantizar una separación clara entre lógica de negocio, datos y presentación. Además, se empleó programación orientada a objetos para el diseño de modelos robustos, funcional para la lógica de negocio, y reactiva con JavaScript puro para interacciones dinámicas en tiempo real.

Dentro de sus funcionalidades más destacadas, el sistema incorpora un módulo de gestión de usuarios personalizado, que diferencia entre Socios, Empresas y Visitantes mediante formularios adaptados a cada tipo. Se ha implementado un sistema de autenticación robusto con middleware que obliga a cambiar la contraseña en el primer ingreso y controla el acceso con sesiones seguras que expiran automáticamente.

La gestión de afiliaciones ha sido digitalizada completamente, permitiendo registrar personas naturales o jurídicas mediante formularios inteligentes que validan automáticamente cédulas, RUC y otros datos. El sistema gestiona la carga de documentos requeridos, verifica su tipo y activa un flujo de aprobación con estados definidos y notificaciones automáticas por correo electrónico.

Otro módulo importante es el sistema de reservas, que permite agendar servicios como salas de reuniones o auditorios mediante un calendario interactivo con control de conflictos de horario. Una vez realizada la reserva, se genera automáticamente un archivo PDF personalizado con la información y el logotipo institucional, y se envía una notificación al usuario con la confirmación correspondiente.

En el ámbito comercial, el sistema permite la gestión de convenios mediante un modelo de categorización inteligente que clasifica los acuerdos por tipo de beneficio (financieros, turísticos, profesionales, etc.). Este módulo calcula automáticamente los porcentajes de descuento, permite gestionar beneficios mediante operaciones CRUD, y genera alertas cuando los convenios están por expirar.

Un componente adicional que mejora la interacción es el chatbot institucional, desarrollado con JavaScript puro. Este asistente virtual responde en tiempo real a preguntas frecuentes sobre servicios, requisitos y procesos, mejorando así la experiencia del usuario al brindar soporte 24/7 de forma automática.

El panel de administración, basado en Django Admin y personalizado con la plantilla Jazzmin, ofrece una interfaz amigable desde la cual el personal puede gestionar contenido, usuarios, convenios y servicios. También incluye métricas y reportes estadísticos en tiempo real que ayudan a tomar decisiones informadas.

El sistema fue construido sobre un stack tecnológico moderno y probado: Django 5.2.3, base de datos SQLite3 para desarrollo y PostgreSQL para producción, y herramientas como Bootstrap 5 y JavaScript para el frontend. Se emplearon librerías especializadas como WeasyPrint para generar PDFs, Pillow para procesamiento de imágenes y Django Forms para validación robusta de formularios.

2. Objetivos

2.1. Objetivo General

- Desarrollar un sistema web integral de gestión empresarial para la Cámara de Comercio de Loja que digitalice y automatice sus procesos administrativos y operativos, mejorando la interacción con socios, la eficiencia interna y la accesibilidad a los servicios institucionales.

2.2. Objetivos Específicos

- Digitalizar el proceso de afiliación de personas naturales y jurídicas, mediante formularios dinámicos, validación automática de datos y carga de documentos con flujo de aprobación en línea.
- Implementar un sistema automatizado para la reserva de servicios institucionales, como salas de reuniones, auditorios y asesorías.
- Desarrollar módulos inteligentes de gestión de convenios y notificaciones, que integren funcionalidades como categorización de beneficios, alertas automáticas, centro de notificaciones y un chatbot para atención en tiempo real.

2.3. Plataforma y tecnologías utilizadas

El prototipo del Sistema de Gestión Empresarial para la Cámara de Comercio de Loja fue desarrollado sobre una arquitectura moderna, escalable y segura, orientada a proporcionar una experiencia de usuario fluida tanto en entornos web como móviles. A continuación, se detalla el conjunto de plataformas y tecnologías específicas empleadas en su desarrollo:

2.3.1. Plataformas de Desarrollo

- Plataforma Web Principal
- Tipo de Aplicación: Aplicación Web Responsive (SPA-like)
- Compatibilidad: Navegadores modernos como Google Chrome, Firefox, Safari y Microsoft Edge
- Arquitectura: Cliente-servidor con renderizado del lado del servidor (SSR)

- Adaptabilidad: Interfaz responsive optimizada para pantallas de escritorio y dispositivos móviles

2.3.2. Tecnologías de Desarrollo

2.3.2.1. Backend

- Lenguaje de Programación: Python 3.8+
- Framework Web Principal: Django 5.2.3, basado en el patrón arquitectónico Modelo-Vista-Template (MVT)
- Gestión de Administración: Django Admin personalizado con Jazzmin para una interfaz moderna y amigable
- Gestión de Formularios: Django Forms para validación robusta y protección CSRF

2.3.2.2. Frontend

- Lenguaje de Programación: JavaScript (ES6+), sin frameworks adicionales (Vanilla JS)
- Framework CSS: Bootstrap 5.3+ para diseño responsive y componentes reutilizables
- Iconografía: Font Awesome 6.x para iconos y símbolos visuales en la interfaz

2.3.2.3. Bases de Datos

Base de Datos para Desarrollo: SQLite3 (base de datos ligera y portable)

Base de Datos para Producción: PostgreSQL 12+, robusta y orientada a producción con soporte ACID

2.3.2.4. Bibliotecas y Dependencias

- WeasyPrint: Para la generación profesional de documentos PDF (como recibos y reportes)
- Pillow (PIL): Procesamiento y optimización de imágenes (redimensionado, conversión de formatos)
- Django Messages: Sistema de mensajes para feedback visual al usuario

2.3.3. Herramientas de Desarrollo y Despliegue

2.3.3.1. Desarrollo Local

- Entorno Virtual: Python venv para aislamiento de dependencias
- Gestión de Dependencias: pip y archivo requirements.txt

- Servidor Local: Django Development Server (runserver) para testing en entorno local

2.3.4. Control de Versiones y CI/CD

- Sistema de Versionamiento: Git con estructura de ramas (desarrollo, staging, producción)
- Plataforma de Repositorio: GitHub/GitLab

2.3.5. Tecnologías de Seguridad

- Autenticación y Autorización: Sistema integrado de Django con hashing seguro de contraseñas y control granular de permisos
- Validación y Sanitización: Validación doble (cliente-servidor) y limpieza automática de datos de entrada
- Gestión de Sesiones: Expiración automática y control de sesiones activas

2.3.6. Experiencia de Usuario e Interfaz

- Interactividad: Validaciones en tiempo real, animaciones y AJAX con Vanilla JavaScript
- Componentes UI: Navbar, cards, formularios, botones y modales personalizados
- Accesibilidad: Cumplimiento de estándares web y navegación intuitiva

2.3.7. Arquitectura Técnica

- Patrón General: MVT (Model-View-Template) de Django
- Flujo de Datos:

- Cliente (navegador) → Nginx (servidor web)
- Nginx → Gunicorn (servidor de aplicaciones)
- Gunicorn → Django (lógica de negocio)
- Django → PostgreSQL (datos)
- Respuesta → Cliente (HTML renderizado + interacciones JS)

3. Requisitos del sistema

Para garantizar una experiencia adecuada y sin interrupciones al utilizar el prototipo del sistema de gestión empresarial para la Cámara de Comercio de Loja, se establecen las siguientes especificaciones técnicas mínimas, para dispositivos de escritorio. Estas especificaciones aseguran la compatibilidad, rendimiento y usabilidad del sistema en su fase actual de desarrollo.

Tabla 1

Dispositivos de Escritorio

Componente	Requisito Mínimo
Sistema Operativo	Windows 10 o superior, macOS 10.14+ o Linux (Ubuntu 20.04+)
Navegador Web Compatible	Google Chrome 90+, Mozilla Firefox 88+, Microsoft Edge 90+, Safari 13+
Procesador (CPU)	Intel Core i3 (4ª generación) o equivalente AMD
Memoria RAM	4 GB
Almacenamiento Libre	200 MB disponibles (archivos temporales y caché)
Resolución de Pantalla	Mínimo 1280x720 px
Conexión a Internet	3 Mbps o superior (recomendado estable)

Nota. En esta tabla se detallan los requisitos mínimos que se deben de tener al momento de correr el aplicativo web.

4. Arquitectura y estructura

La arquitectura del sistema está basada en un enfoque cliente-servidor con arquitectura en tres capas: presentación, lógica de negocio y persistencia de datos. El sistema utiliza tecnologías web modernas, lo que permite su ejecución en navegadores sin necesidad de instalar software adicional, y está diseñado para facilitar su escalabilidad y mantenimiento.

4.1. Capas de la Arquitectura

4.1.1. Capa de Presentación (Frontend)

- Desarrollada con **HTML5, CSS3, Bootstrap y JavaScript**.
- Interfaz amigable, responsive y accesible desde cualquier dispositivo.
- Los formularios permiten:
    - Registro de usuarios
    - Reserva de servicios
    - Visualización de convenios
    - Generación y descarga de documentos PDF

4.1.2. Capa de Lógica de Negocio (Backend)

- Implementada en **Python usando el framework Django**.
- Procesa todas las solicitudes del cliente (usuario).
- Contiene:
    - Validaciones de entrada
    - Gestión de sesiones y permisos
    - Generación de archivos PDF con WeasyPrint

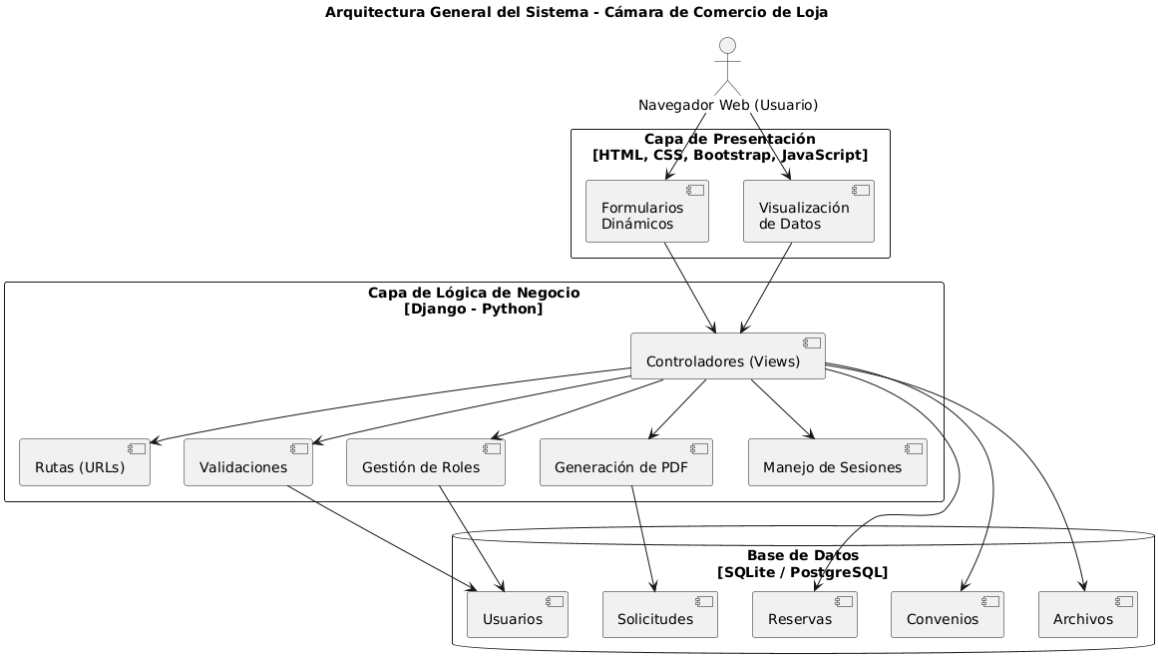
- Control de reservas, afiliaciones y convenios

4.1.3. Capa de Persistencia (Base de Datos)

- Base de datos: **SQLite** (modo prototipo) o **PostgreSQL** (modo producción).
- Contiene tablas para:
  - Usuarios y roles (personas naturales, jurídicas, administradores)
  - Solicitudes de afiliación
  - Reservas de servicios
  - Convenios institucionales
  - Notificaciones

Figura 1

Arquitectura General del Sistema



Nota. La figura representa el como el usuario accede al apliocativo y como el aplicativo se relaciona entre sí.

5. Funcionalidades principales

5.1. **Módulo de Autenticación y Roles**

- Inicio de sesión seguro para usuarios registrados.
- Registro de nuevos usuarios (personas naturales o jurídicas).
- Gestión de roles (administrador, socio, asistente, etc.).
- Restricción de funcionalidades según el rol.

5.2. Módulo de Afiliación

- Formularios dinámicos de afiliación para personas naturales y jurídicas.
- Subida de documentos requeridos en PDF o imagen (cédula, solicitud).
- Registro automático de la solicitud en la base de datos.
- Visualización del estado de solicitud (en revisión, aprobada, rechazada).

5.3. Módulo de Reserva de Servicios

- Formulario para reservar salas de reuniones, auditorios o asesorías.
- Selección de fecha y hora con calendario dinámico.
- Generación automática de comprobante de reserva en PDF.
- Confirmación de reserva vía correo electrónico o notificación.
- Consulta de historial de reservas realizadas.

5.4. Módulo de Gestión de Convenios

- Visualización de convenios disponibles y sus beneficios.
- Filtro por categoría (salud, educación, servicios, etc.).

- Registro de uso de convenios por parte del socio.

5.5. Módulo de Notificaciones y Comunicaciones

- Notificaciones automáticas por acciones del sistema (reserva confirmada, solicitud aprobada, etc.).
- Centro de notificaciones para cada usuario.
- Envío de mensajes o anuncios institucionales por parte del administrador.

5.6. Módulo de Asistencia Virtual

- Chatbot de atención automatizada para resolver preguntas frecuentes.
- Derivación a contacto humano en caso de preguntas avanzadas.

5.7. Módulo Administrativo (para gestores de la Cámara)

- Panel de control con resumen de estadísticas (afiliaciones, reservas, convenios).
- Gestión de usuarios y solicitudes.
- Administración de convenios (alta, edición, baja).

6. APIs y servicios externos

6.1. Gmail SMTP – Servicio de Correo Saliente

El sistema utiliza el **servicio SMTP de Gmail** como proveedor externo para el **envío automático de correos electrónicos** desde el sistema web. Este servicio permite notificar a los usuarios sobre eventos relevantes como la aprobación de afiliaciones, confirmaciones de registro, reservas de servicios y recordatorios institucionales.

6.1.1. API/Protocolo Utilizado

- **Protocolo**: SMTP (Simple Mail Transfer Protocol)
- **Servidor SMTP**: smtp.gmail.com
- **Puerto**: 587 (con TLS)
- **Autenticación**: Usuario y contraseña de aplicación (OAuth2 opcional)

6.1.2. Librería de integración

- Utiliza el **backend SMTP nativo de Django**:

  python

  CopiarEditar

  EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

6.1.3. Credenciales

- **Usuario SMTP**: loja2002ecuador@gmail.com
- **Contraseña de aplicación**: Se genera desde la cuenta de Gmail (con autenticación de dos factores activada)
- **DEFAULT_FROM_EMAIL**: 'Cámara de Comercio <loja2002ecuador@gmail.com>'

6.1.4. Configuración y Requisitos de Integración

6.1.4.1. Requisitos previos

- Cuenta de Gmail válida con autenticación en dos pasos habilitada.

- Generación de contraseña de aplicación desde el panel de seguridad de Google.
- Acceso a modificar el archivo settings.py en Django.

6.1.4.2. Configuraciones en settings.py

python

CopiarEditar

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = os.environ.get('EMAIL_HOST_USER')
EMAIL_HOST_PASSWORD = os.environ.get('EMAIL_HOST_PASSWORD')
DEFAULT_FROM_EMAIL = os.environ.get('DEFAULT_FROM_EMAIL')
```

6.1.4.3. Seguridad recomendada

- Uso de variables de entorno para ocultar credenciales.
- Rotación periódica de contraseñas de aplicación.
- Limitación de acceso al entorno de producción mediante políticas de seguridad de Django y Google.

7. Equipo de desarrollo

Tabla 2

Equipo de desarrollo

#	Nombres	Rol	Responsabilidades Asignadas
1	Luis Fernando León Tène	**Desarrollador Backend - Modelos y Base de Datos** *Arquitecto de Datos y Modelos*	- Diseño y desarrollo de modelos de datos - Configuración de la base de datos - Migraciones y esquemas - Relaciones entre entidades - Validaciones en modelo
2	Omer Alexis Benitez Cabrera	**Desarrollador Backend - Vistas, Formularios y URLS** *Lógica de Negocio y Controladores*	- Desarrollo de vistas (views) - Creación de formularios - Configuración de URLs - Lógica de autenticación/autorización - Integración con servicios externos
3	Lenin Andrés Cuenca Cuenca	**Desarrollador Frontend - Templates HTML y CSS** *Interfaz y Experiencia de Usuario*	- Desarrollo de templates HTML - Diseño y estilos CSS - JavaScript interactivo - Integración con Bootstrap

Nota. Descripcion y resposabilidades del equipo de desarrollo.

8. Versiones y evolución del prototipo

8.1. Versión 1.0 – Versión Inicial (MVP - Producto Mínimo Viable)

- **Fecha de lanzamiento:** 20/05/2025
- **Características principales:**
    - Autenticación de usuarios (registro e inicio de sesión).
    - Gestión básica de formularios (creación y envío).
    - Almacenamiento de datos en base de datos local (SQLite).
    - Interfaz funcional (HTML + CSS + Bootstrap).
- **Cambios clave:** Primera integración completa entre frontend, lógica de negocio y base de datos.

8.2. Versión 1.1 – Ajustes y Correcciones

- **Fecha de lanzamiento:** 03/06/2025
- **Mejoras:**
    - Validaciones mejoradas en los formularios.
    - Corrección de errores en rutas y visualización de datos.
    - Mejora en el diseño.
- **Cambios clave:** Refinamiento del flujo de usuario y experiencia visual.

8.3. Versión 2.0 – Versión Ampliada

- **Fecha estimada de lanzamiento:** 01/07/2025
- **Nuevas funcionalidades previstas:**
    - Integración de generación de reportes en PDF.
    - Sistema de roles (administrador, usuario, moderador).
    - Notificaciones por correo electrónico.
- **Cambios clave esperados:** Escalabilidad y robustez del sistema para ambientes reales.

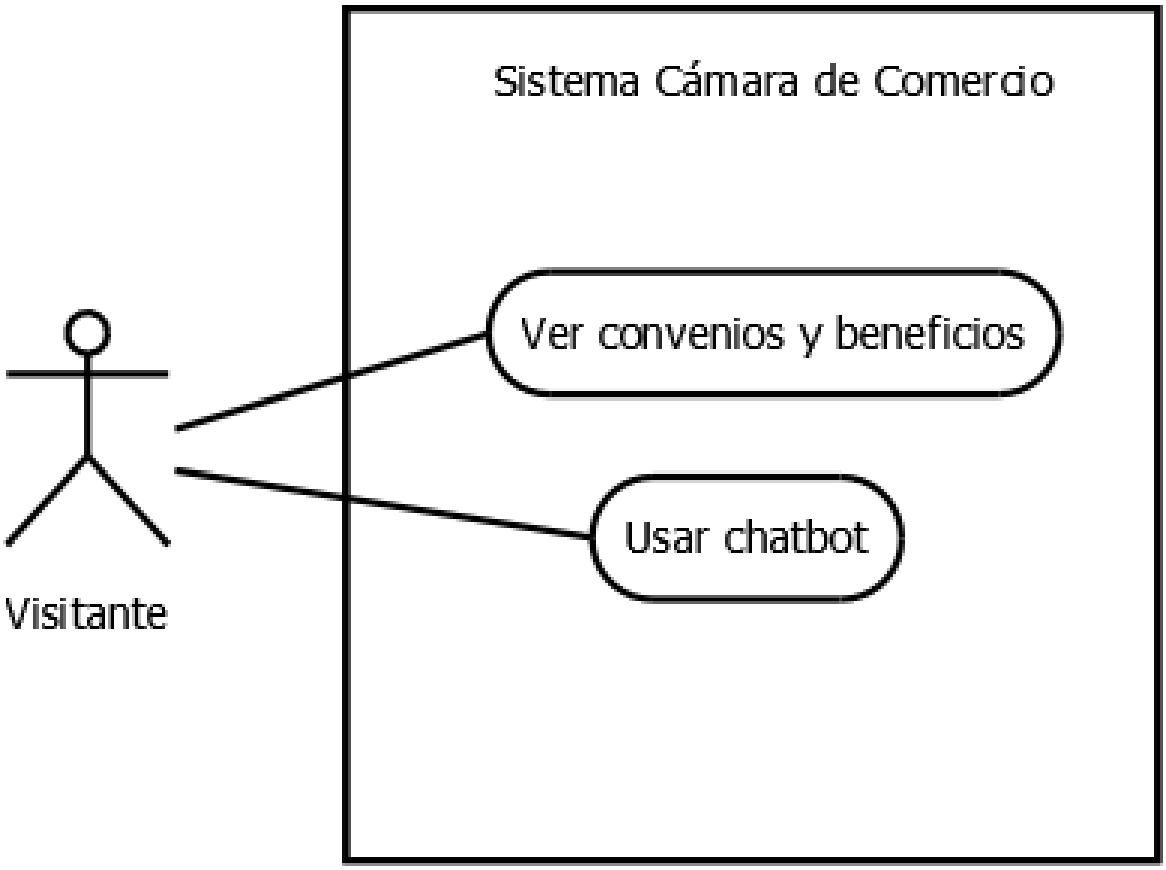
8.4. Evolución Futura del Prototipo

- **Versión 3.0 (Propuesta):**
    - Panel de estadísticas con visualización gráfica.
    - Dashboard de administración avanzada.
    - Internacionalización y soporte multilingüe.
    - API REST para integración con sistemas externos.
    - Mejora en accesibilidad y cumplimiento de estándares WCAG.

9. Anexos

Figura 2

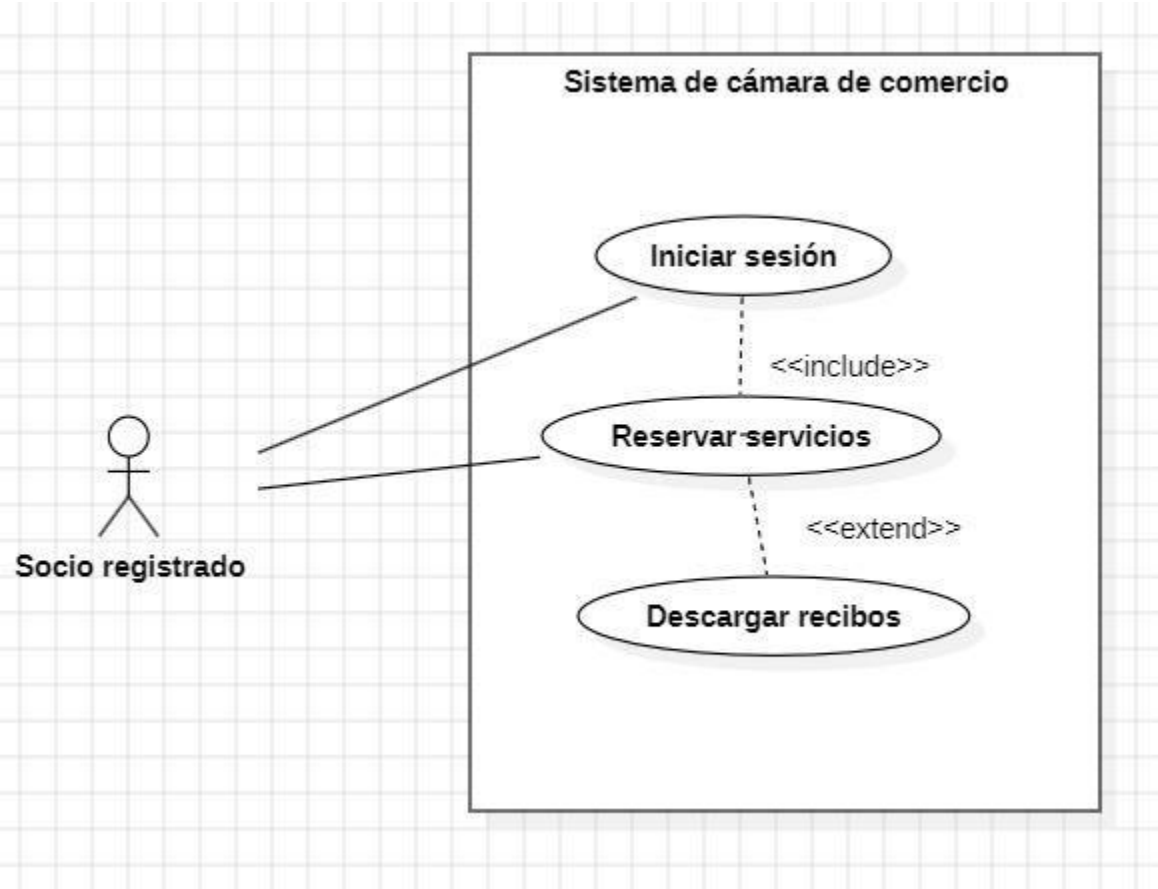
Diagrama de Caso de Uso



Nota. La imagen muestra el caso de suso del Usuario visitante y lo que va a poder hacer en el aplicativo.

Figura 3

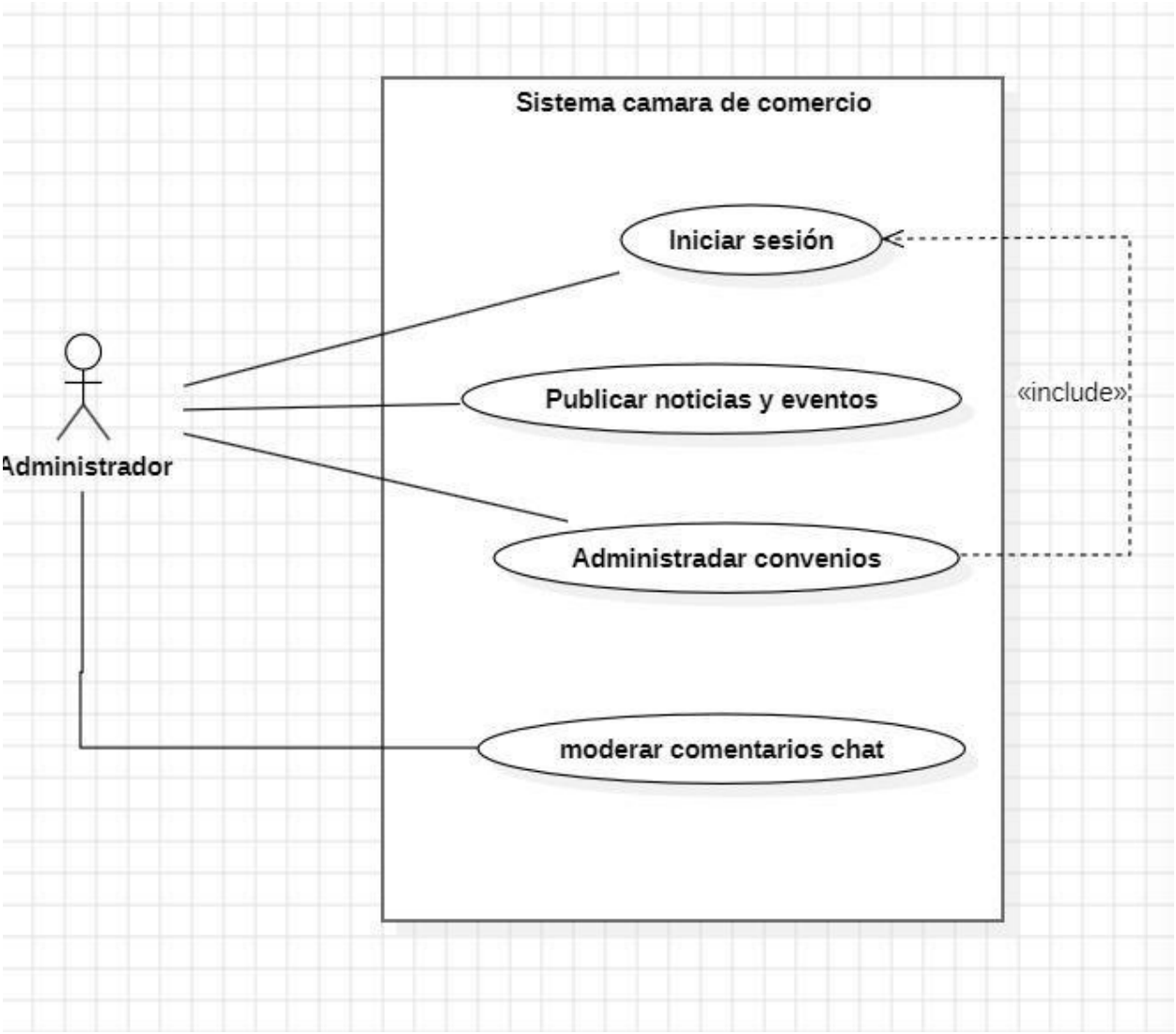
Diagrama de Caso de Uso



Nota. La imagen muestra el caso de uso del socio una vez registrado y a lo que va a poder acceder al aplicativo.

Figura 4

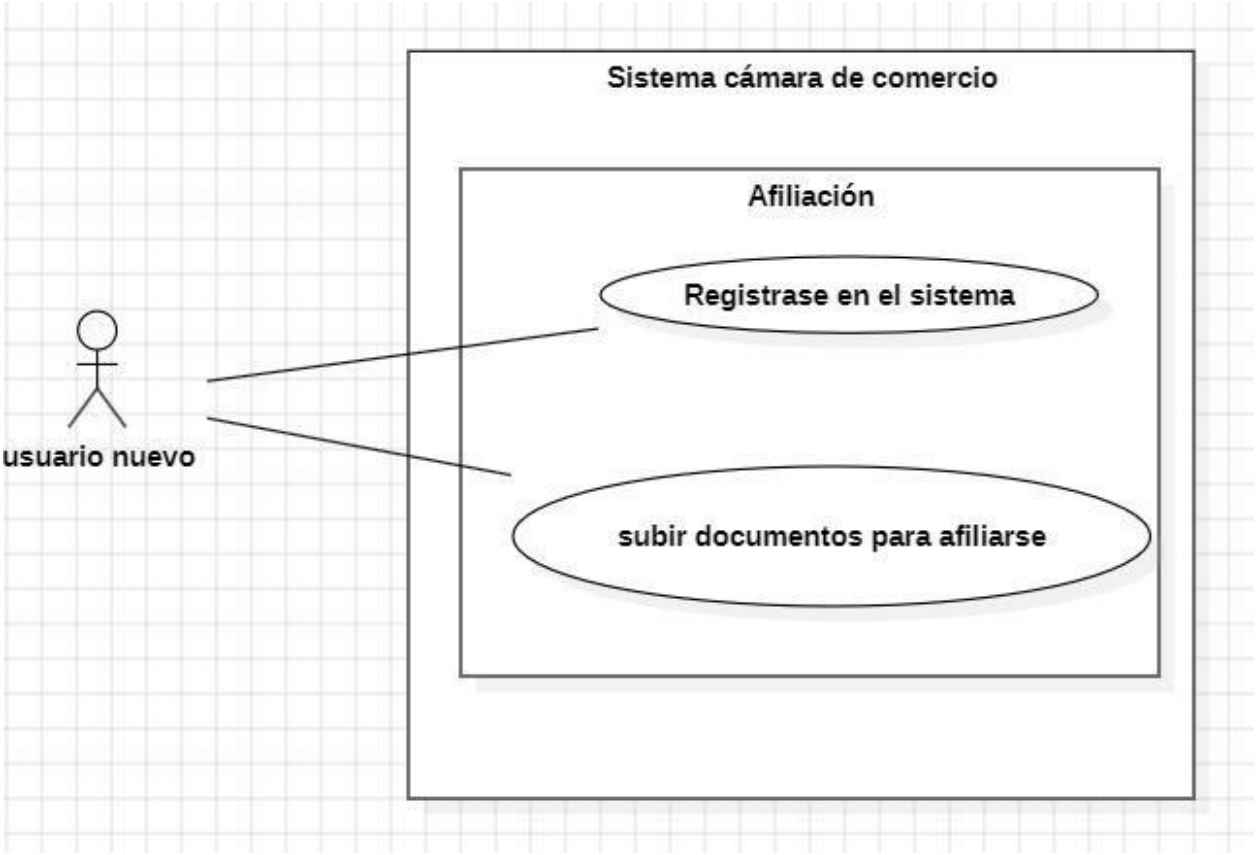
Diagrama de Caso de Uso



Nota. La imagen muestra el caso de uso del administrador y todo lo que va a poder hacer en el aplicativo.

Figura 5

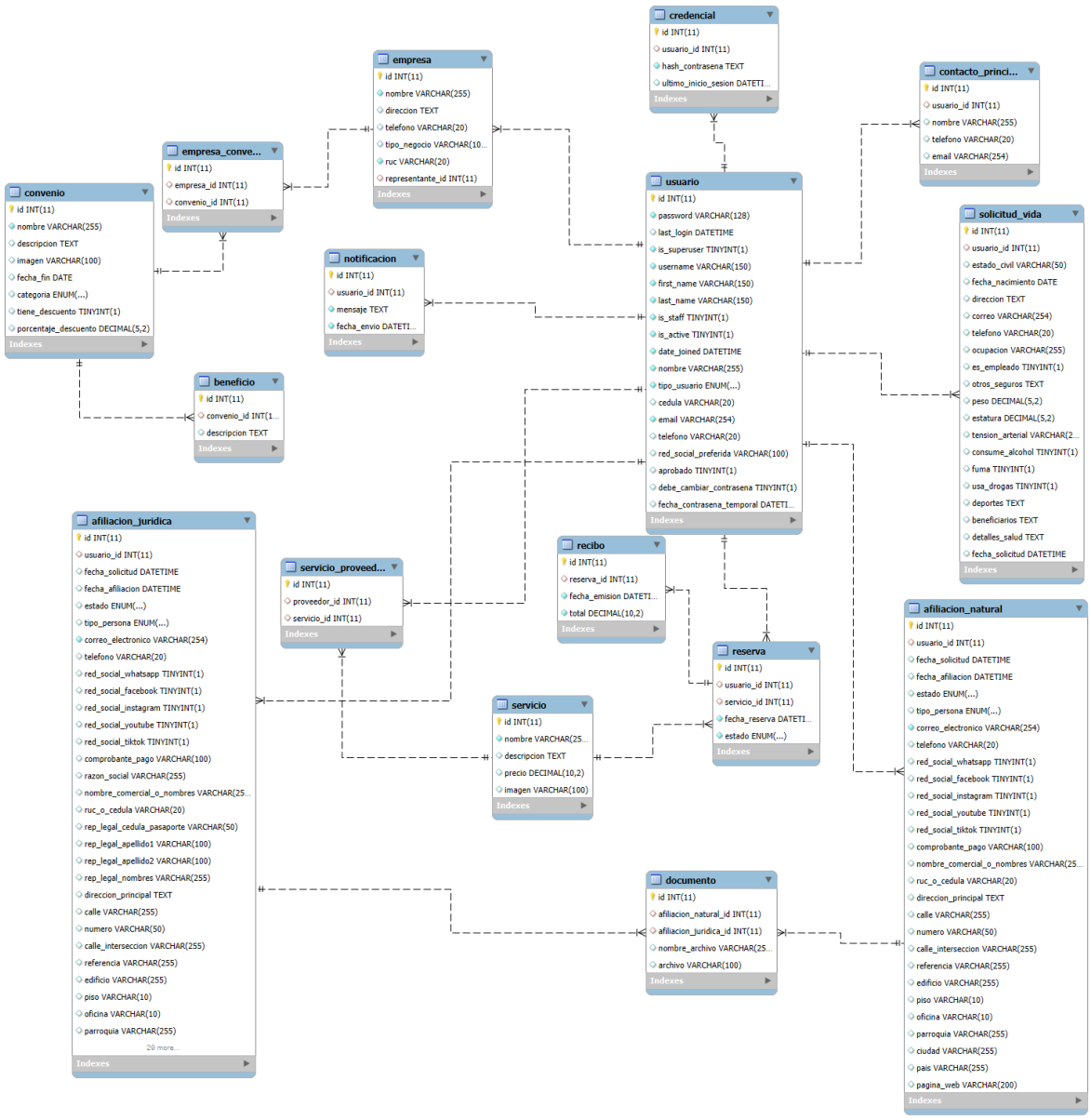
Diagrama de Caso de Uso



Nota. La imagen muestra el diagrama de caso de uso del usuario nuevo que se registró y los que va a poder hacer dentro del aplicativo

Figura 6

Base de datos



Nota. La figura muestra todas las entidades que conforman la base de datos y sus respectivas relaciones.