INSTITUTO TECNOLÓGICO SUPERIOR

SUDAMERICANO

QUITO - ECUADOR

ESCUELA DE

DESARROLLO DE SOFTWARE

PROYECTO DE TITULACIÓN

DESARROLLO DE UN SISTEMA DE CONTROL DE CONDOMINIO PARA EL EDIFICIO PLAZA 10 UBICADO EN EL DISTRITO METROPOLITANO DE QUITO

AUTOR: SAINZ RIVERON ALCIDES

COORDINADOR: MSc. FABRIZIO VILLASIS

AGOSTO 2024

QUITO – ECUADOR

AUTORÍA

Yo, Alcides Sainz Riverón, portador de la cédula de ciudadanía No. 1756597983, declaro bajo juramento que el trabajo aquí descrito, es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional y que he consultado e investigado en base a las referencias bibliográficas que se incluyen en este documento. Esta investigación no contiene plagio alguno y es resultado de un trabajo serio desarrollado en su totalidad por mi persona.

_____

Alcides Sainz Riveron

CERTIFICACIÓN

Una vez que se ha culminado la elaboración del proyecto de titulación cuyo tema es: "Desarrollo de un sistema de control de condominio para el edificio Plaza 10 ubicado en el Distrito Metropolitano de Quito", certifico que el mismo se encuentra habilitado para su defensa pública.

MSc. Fabrizio Villasis Chiriboga
Coordinador de la Escuela de
Desarrollo de Software
Instituto Superior Tecnológico Sudamericano Quito

CERTIFICACIÓN

Por medio del presente certifico que el señor Alcides Sainz Riveron, ha realizado y concluido su trabajo de titulación, cuyo tema es: “Desarrollo de un sistema de control de condominio para el edificio Plaza 10 ubicado en el Distrito Metropolitano de Quito”, para obtener el título de Tecnólogo en Desarrollo de Software, bajo mi tutoría.

_____

MSc. Fabrizio Vicente Villasis Chiriboga
Director del Proyecto de Titulación

AGRADECIMIENTOS

Deseo expresar mi más profundo agradecimiento a todas las personas que han contribuido a la realización de esta tesis. Sin su apoyo y compañía, este logro no habría sido posible.

Quiero agradecer a mi familia, cuyo amor y apoyo incondicional han sido mi mayor fuente de fortaleza. A mis padres, hermana y seres queridos, gracias por creer en mí, por sus palabras de ánimo y por estar siempre a mi lado, especialmente en los momentos más desafiantes. Sus sacrificios y su fe en mí han sido fundamentales para alcanzar esta meta.

Gracias por ser parte de este logro.

DEDICATORIA

A mis padres, por su amor incondicional, su sacrificio y por enseñarme el valor del esfuerzo y la perseverancia. Sin su guía y apoyo, este logro no habría sido posible.

A mi hermana, por ser una fuente constante de inspiración y por su inquebrantable apoyo en cada etapa de mi vida.

A mi novia, por su amor, paciencia y por estar siempre a mi lado, brindándome su apoyo y comprensión en todo momento. Tu compañía ha sido fundamental para superar cada desafío.

A mis suegros, por acogerme en su familia y por su apoyo incondicional. Sus palabras de aliento y su confianza en mí han sido un gran respaldo.

Gracias a todos.

RESUMEN

El trabajo de titulación se enfoca en el desarrollo de un sistema de control de condominio para el edificio Plaza 10, ubicado en el Distrito Metropolitano de Quito. Su objetivo general es desarrollar e implementar dicho sistema para asegurar el control de condominio del referido inmueble, con el fin de mejorar la gestión administrativa. Además, se plantean como objetivos específicos identificar y enunciar los aspectos teóricos relacionados con las herramientas necesarias para el diseño e implementación del proyecto prototipo para la gestión de estos inmuebles. Para ello, se elabora un marco teórico que sustenta la definición de condominio, sus clases y ventajas, entre otros aspectos. Unido a esto, se analiza teóricamente el desarrollo web a partir del estudio de ASP.NET Core, C#, Entity Framework, SQL Server, API, GitHub, entre otros.

Asimismo, se encuestan a 163 personas del edificio para identificar los principales problemas existentes y, con base en estos hallazgos, desarrollar el sistema de control para la administración del mismo. Todo lo anterior se lleva a cabo mediante la metodología de desarrollo MVC (Model View Controller), lo que da como resultado la creación de un software que resuelve las exigencias más inmediatas mediante un conjunto de funcionalidades, como: reservas de áreas comunes, consulta de pagos de alícuotas, registro de visitas, entre otras. Se concluye que el sistema permitirá asegurar una gestión administrativa eficiente, aplicable a este modelo de vivienda, y una adecuada convivencia entre sus residentes.

Palabras clave: ASP.NET Core, condominio, desarrollo web y sistema de control de condominios edificio Plaza 10.

ABSTRACT

The degree project focuses on the development of a condominium control system for the Plaza 10 building, located in the Metropolitan District of Quito. Its general objective is to develop and implement this system to ensure the condominium control of the property, with the aim of improving administrative management. Additionally, the specific objectives include identifying and outlining the theoretical aspects related to the necessary tools for the design and implementation of the prototype project for the management of these properties. For this purpose, a theoretical framework is developed, which supports the definition of a condominium, its types and advantages, among other aspects. Furthermore, the web development process is theoretically analyzed based on the study of ASP.NET Core, C#, Entity Framework, SQL Server, API, GitHub, among others.

Additionally, 163 residents of the building are surveyed to identify the main existing problems and based on these findings, to develop the control system for the administration of the property. All of this is carried out using the MVC (Model View Controller) development methodology, resulting in the creation of software that addresses the most immediate needs through a set of functionalities such as: common area reservations, consultation of alimony payments, visitor registration, among others. It is concluded that the system will ensure efficient administrative management, applicable to this housing model, and promote proper coexistence among its residents.

Keywords: ASP.NET Core, condominium, web development and Plaza 10 building condominium control system.

ÍNDICE

ÍNDICE DE FIGURAS

1. Introducción

Los condominios representan un estilo de vida cada vez más común en numerosas ciudades alrededor del mundo. En Quito, la expansión y modernización de la ciudad han dado lugar a la construcción de diversos complejos residenciales, como es el caso del edificio Plaza 10. Para que estos inmuebles funcionen de manera óptima, es fundamental contar con una administración eficiente que se encargue de tareas como el control de pagos de las cuotas, la reserva adecuada de áreas comunes, el registro de visitantes y otras actividades que contribuyan al buen funcionamiento del lugar.

En esa línea, este trabajo de investigación tiene como fin resolver las problemáticas del inmueble mencionado, de maneras que se administre y gestione adecuadamente. Por ello, el tema de investigación se enfoca al "Desarrollo de un sistema de control de condominio para el edificio Plaza 10 ubicado en el Distrito Metropolitano de Quito". Para llevarlo a cabo se formuló como objetivo general, desarrollar e implementar un sistema integral de control de condominio para el referido inmueble para mejorar la gestión administrativa, reservas de áreas comunes, consulta de pagos de alícuotas, registro de visitas, entre otras funciones para lograr la eficiencia en la administración y mejorar la calidad de vida de sus habitantes.

Igualmente, se trazaron como objetivos específicos identificar y enunciar los aspectos teóricos que se relacionan con las herramientas necesarias para el diseño e implementación del proyecto prototipo para la gestión de condominios. También, analizar las necesidades del edificio Plaza 10 en cuanto a la gestión de condominios, identificando las funcionalidades claves que debe ofrecer el sistema. Asimismo, desarrollar un sistema de control de condominio para el edificio Plaza 10 con base a la metodología de desarrollo MVC. Por último, realizar las pruebas necesarias del sistema para verificar su correcto funcionamiento.

Este trabajo está estructurado en seis partes. La primera, se dedica a los aspectos introductorios relacionados con la problemática, la justificación del tema, sus antecedentes y objetivos. Otra de las temáticas que se tratan es el marco teórico que sirve de sostén al sistema de condominio a partir del estudio de su definición, clases, ventajas al igual que la importancia de aplicar en este contexto la tecnología, en especial una aplicación web.

Por otro lado, dentro del marco teórico, se revisa lo referente a los tipos de *software* que se aplican en la administración de condominio. Asimismo, se enfatiza en el desarrollo web, en los lenguajes de programación, en especial C#. También se estudian *los frameworks*, acerca de los *hostings*, las bases de datos, las API, entre otros aspectos que luego se aplican al sistema.

Luego se desarrolla el proyecto correspondiente donde se detallan los objetivos específicos cumplidos. Específicamente, los resultados de la encuesta que exponen las necesidades que tienen los residentes del edificio Plaza 10 de contar con un sistema para administrar el condominio adecuadamente. Asimismo, se detallan las tecnologías utilizadas al igual que el comportamiento general del sistema. De igual modo se muestran cada uno de los componentes de la interfaz, sus funcionalidades. Por último, las pruebas aplicadas para validarlo y ponerlo en práctica.

2. Justificación

La problemática existente en el del edificio Plaza 10 radica en que la administración y los residentes no cuentan con un sistema de condominios que permita mejorar y optimizar la administración de este y ayude a mejorar la calidad de vida de los residentes. Dicha cuestión obstaculiza y le resta rapidez al desarrollo de las tareas propias para gestionar. Igualmente impacta de forma negativa en la administración de este inmueble. Por ello, este proyecto de sistema de control permitirá brindar una solución efectiva para optimizar y dar mayor eficacia a las tareas que se derivan de la administración.

Por lo anterior, es importante crear e implementar un sistema de control de condominio porque este ayudaría al personal de la administración del edificio Plaza 10, a su guardianía y habitantes. Con este se permitirá una mayor integración entre el aparato administrativo y los residentes, los que podrán ser parte de diversas actividades y obligaciones que se debe de cumplir. Asimismo, entre las tareas que se realizaran hacer mediante el sistema están: la recaudación de cuotas, registro de visitas, reservas de áreas y configuración de la directiva del condominio. Todo ello permitirá una gestión rápida y eficiente de los recursos de dicho edificio en beneficio de sus 80 departamentos.

Igualmente, la importancia de investigar y desarrollar un sistema de control de condominio en el edificio Plaza 10 se sustenta en varias razones fundamentales. En primer lugar, la ineficiencia inherente a la gestión manual o el uso de herramientas informales en la administración de condominios conlleva a notables ineficiencias en aspectos clave como la recaudación de cuotas, el mantenimiento de áreas comunes, la comunicación entre propietarios y la toma de decisiones colectivas. Estas deficiencias impactan directamente en la calidad de

vida de los residentes y pueden generar conflictos dentro de la comunidad, lo que resalta la necesidad de investigar soluciones más efectivas.

Otra razón para el desarrollo de este proyecto se sustenta en que la implementación de un sistema de control de condominio eficiente no solo responde a la necesidad de mejorar la calidad de vida de los habitantes, al garantizar el funcionamiento óptimo de las instalaciones y servicios, sino también a la urgencia de ahorrar recursos. Un sistema bien diseñado y adaptado a las necesidades específicas de esta comunidad permitiría un uso más eficiente de los recursos disponibles, reduciendo gastos innecesarios en mantenimiento y servicios, lo que se traduce en beneficios económicos y ambientales.

Otro aspecto que respalda la importancia de este proyecto es la capacidad de resolver conflictos de manera efectiva. La implementación de herramientas de comunicación eficaces y la toma de decisiones basada en datos objetivos contribuirían significativamente a la reducción de conflictos entre propietarios, fomentando un ambiente transparente, colaborativo y armonioso en el condominio.

Del mismo modo, impacta de forma positiva desde el punto de vista académico. Esto a partir de que, la implementación de un sistema de control de condominio ofrece oportunidades de aprendizaje y desarrollo profesional para el personal de administración. Con ello se adquieren habilidades en el uso de tecnologías de gestión y en la mejora de procesos administrativos. Asimismo, para el Instituto, brinda un campo de estudio e investigación en el ámbito de la gestión de condominios y la optimización de recursos en entornos residenciales.

3. Antecedentes

Para revisar los antecedentes, es necesario señalar que la aplicación de sistemas de condominio en la actualidad ha experimentado importantes avances y cambios significativos. Anteriormente, la gestión de estos inmuebles se realizaba principalmente de forma manual o mediante herramientas informales como hojas de cálculo o documentos en papel. Sin embargo, con el avance de la tecnología y la digitalización de procesos, se han desarrollado sistemas de control para ello, cada vez más sofisticados y eficientes.

En relación con la temática de estudio, se han realizado varias investigaciones, fundamentalmente enfocadas en sistemas de control para inmuebles. Por ejemplo, la Universidad Técnica del Norte se llevó a cabo un estudio titulado "Desarrollo de un sistema web; fortalecimiento de los procesos de gestión administrativa y financiera; condominio Solar del Río; Ciudad De Ibarra; Microsoft Azure". En este trabajo se implementó, en Microsoft Azure, el sistema para facilitar su acceso desde cualquier lugar con conexión a Internet, aprovechando la facilidad de configuración de esta plataforma. Se desarrolló utilizando .NET Core y Microsoft SQL Server, aplicando la metodología de desarrollo de aplicaciones web MVC para una gestión eficiente del proyecto (Coronado, 2019).

Por otro lado, en la ESPE se desarrolló la investigación titulada "Desarrollo de un sistema software multiplataforma basado en un modelo de gestión administrativa enmarcado en la arquitectura REST y REDUX para la optimización de la administración del conjunto habitacional "Oriental". En ella e trazaron como objetivos el desarrollo de un sistema software multiplataforma. Unido a ello, construir las bases teóricas acerca de los modelos de gestión administrativa que se utilizan en el condominio diagnosticar las necesidades para luego desarrollar el sistema correspondiente (Montalvo & Paredes, 2023).

También, en la Universidad Católica de Santiago de Guayaquil se estudió sobre el "Manejo óptimo de la gestión económica y administrativa de los condominios que se encuentran bajo el régimen de propiedad horizontal en la Ciudad De Guayaquil". Dicha investigación se enfocó más en controles de tipo económico que en los servicios que brindan los condominios en sí. No obstante, este estudio constituye un aporte desde el punto de vista teórico y de las funcionalidades de tipo económico (Buri, 2021).

Como se aprecia, los estudios dirigidos al desarrollo de un sistema de control de condominio son escasos en el país. Además, esta investigación en particular se dirige a un edificio en específico, que es el Plaza 10 que tiene sus propias características y necesidades.

Lo anterior, muestra la evolución de la aplicación de sistemas de condominio en la actualidad, que se caracteriza por la digitalización, la automatización, la optimización de recursos, la mejora de la experiencia del usuario y la incorporación de tecnologías avanzadas para una gestión más eficiente y transparente de los condominios.

4. Objetivos

4.1. Objetivo General

Desarrollar e implementar un sistema integral de control de condominio para el edificio Plaza 10 para mejorar la gestión administrativa, reservas de áreas comunes, consulta de pagos de alícuotas, registro de visitas, entre otras funciones para lograr la eficiencia en la administración y mejorar la calidad de vida de sus habitantes.

4.2. Objetivos Específicos

- Identificar y enunciar los aspectos teóricos que se relacionan con las herramientas necesarias para el diseño e implementación del proyecto prototipo para la gestión de condominios.
- Analizar las necesidades del edificio Plaza 10 en cuanto a la gestión de condominios, identificando las funcionalidades claves que debe ofrecer el sistema.
- Desarrollar un sistema de control de condominio para el edificio Plaza 10 con base a la metodología de desarrollo web MVC.
- Realizar las pruebas necesarias del sistema para verificar su correcto funcionamiento.

5. Marco Teórico

5.1. Condominios

En la actualidad el estilo de vida y de las viviendas se ha modificado a la par del desarrollo económico, tecnológico y de las ciudades en general (Hidalgo, 2019). En ese orden, los condominios y las urbanizaciones cerradas constituyen un nuevo modelo de construcción en el ámbito urbano, lo que se manifiesta en la ciudad de Quito. Ello conduce a cambios significativos en la manera en que la población y las actividades económicas crecen y se distribuyen dentro del espacio, las que a su vez exigen de ciertos servicios para garantizar el confort de sus habitantes y el funcionamiento adecuado (Meyer & Jurgen, 2020).

En ese orden, según la Real Academia de la Lengua Española el término "condominio" se refiere a una forma de propiedad en la que múltiples personas comparten esta y a su vez, tienen responsabilidad sobre una unidad, que es su vivienda y sobre el edificio. Además, tiene derechos, específicamente a sus áreas comunes, las que deben ser objeto de cuidado por todos para asegurar su disfrute (Real Academia de la Lengua Española, 2018).

En relación con el condominio se afirma:

Un condominio o copropiedad son los terrenos o construcciones (por ejemplo, un conjunto de viviendas), donde coexisten bienes que son de todos y bienes que son de cada propietario. Por lo general corresponden a edificios de departamentos o de casas que se han construido sobre un terreno común (Serrano, 2017,p.3).

En la cita anterior se destaca la existencia de un grupo de viviendas que conforman el condominio donde el propietario, es dueño de su departamento y a su vez, es comunero de los espacios para el uso de todo. Esto se relaciona con prácticas de organización que lleva a cabo

la administración de este tipo de inmuebles donde se disfruta de espacios colectivos como gimnasios, piscinas, sala comunal, estacionamientos, entre otros (Giglia, 2016). Lo que da un significado colectivo al hogar, a la comunidad. Además, el condominio se rige por normas jurídicas, las que deben ser cumplidas a cabalidad.

En el marco del condominio, se pagan cuotas que son contribuciones financieras que los propietarios de unidades deben abonar, normalmente de manera mensual (Montoya, 2016). Estas se destinan a cubrir los gastos de mantenimiento, reparaciones, servicios comunes como: jardinería, limpieza de áreas comunes, seguridad, iluminación, entre otros. Su monto puede variar dependiendo del tamaño de la unidad, las instalaciones y los servicios que ofrece el condominio, así como de las necesidades de mantenimiento y operación. Dicho monto se define por la Asamblea de Copropietarios.

5.1.1. Clases de condominios

Corresponde exponer que existen diferentes clases de condominio de acuerdo con la estructura que tengan, su empleo y el régimen legal a que este sujeto (Choto, 2018). A continuación, se detallan algunos de ellos, especialmente los más comunes.

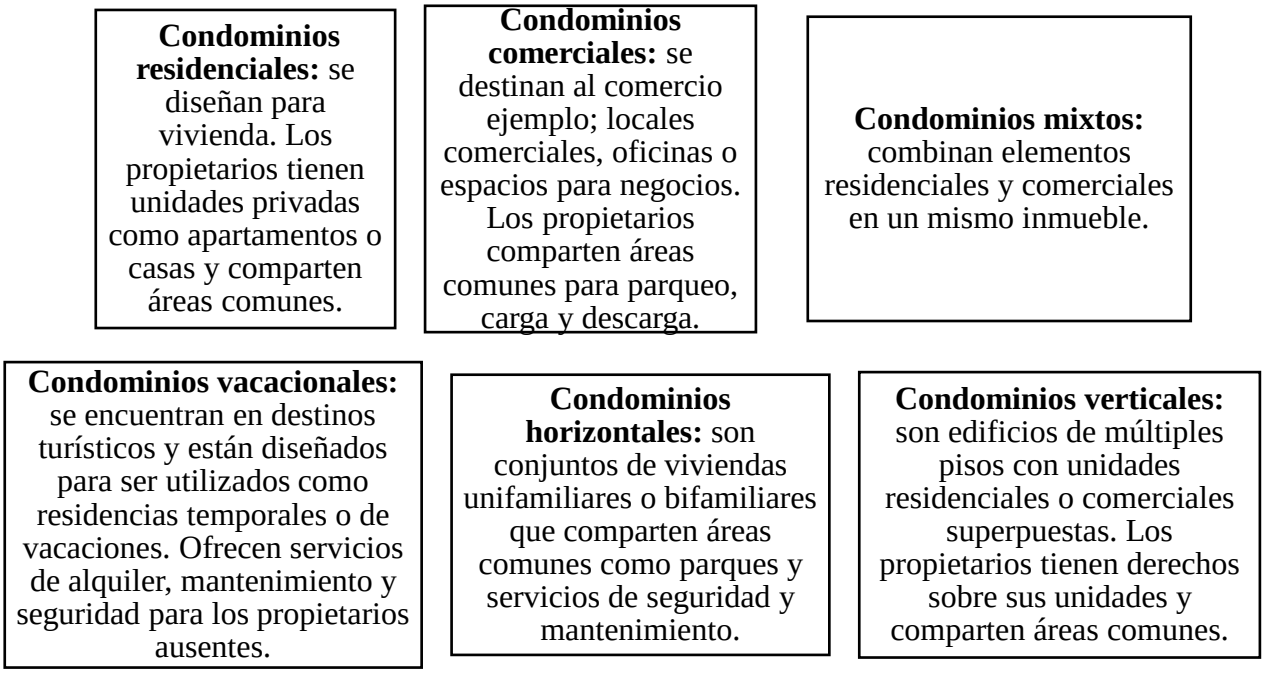


Figura 1. Clases de condominio.
Fuente: (Choto, 2018)

5.1.2. Ventajas del condominio

Los condominios juegan un papel importante en la vida urbana moderna al proporcionar un ambiente seguro, cómodo y comunitario para sus residentes, además de contribuir al desarrollo sostenible de las ciudades al optimizar el uso de recursos y servicios compartidos. Sus principales ventajas se detallan a continuación:

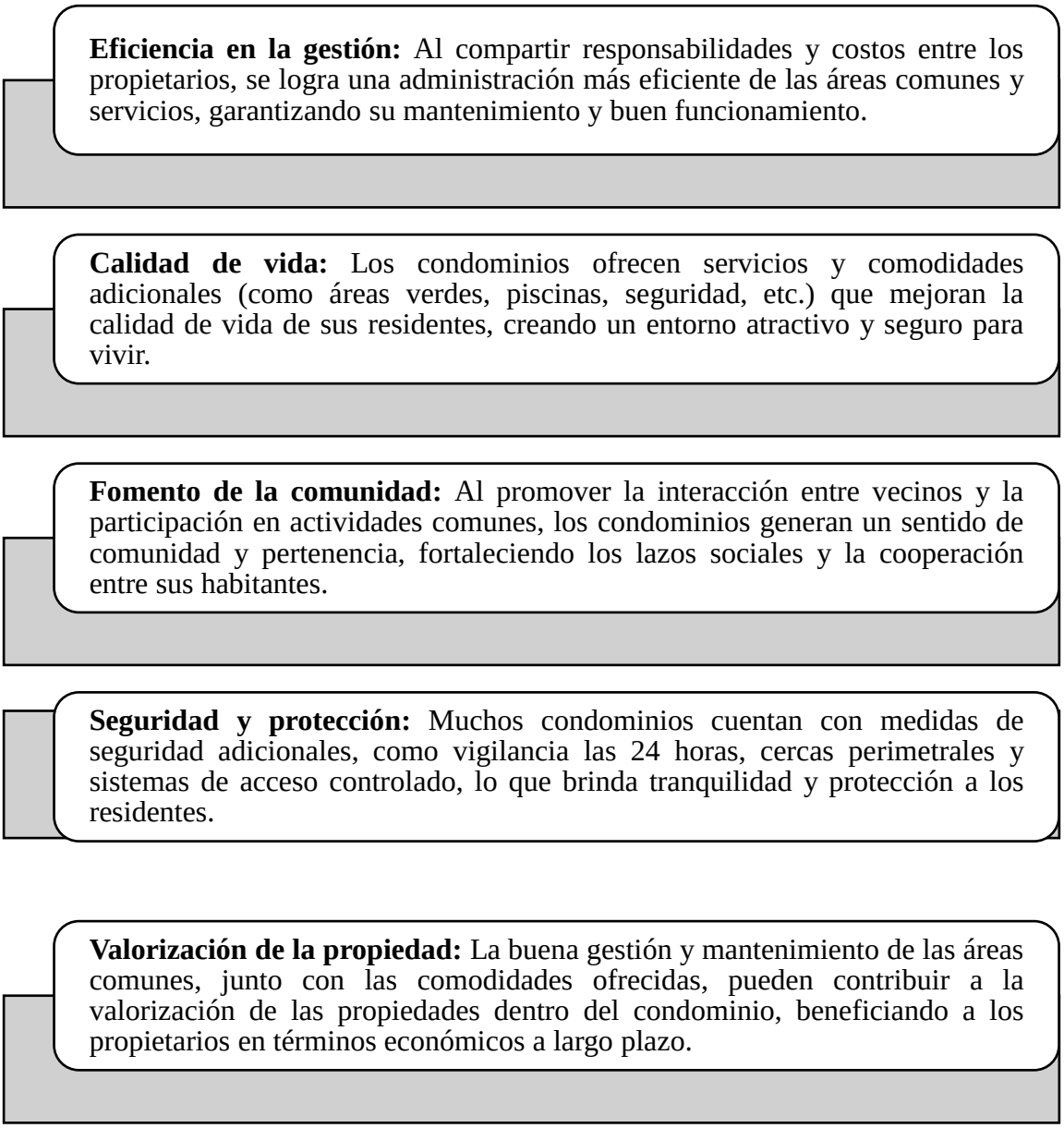


Figura 2. Ventajas del condominio.
Fuente: (Vázquez, 2017)

5.1.3. Roles y Responsabilidades

El condominio está conformado por tres órganos de administración. El primero, es la Asamblea de Copropietarios que es el máximo órgano de gobierno de este modelo de vivienda y se encarga de aprobar el presupuesto, resolver disputas entre copropietarios, establecer los reglamentos internos y tomar las decisiones más importantes (Ley de Propiedad Horizontal, 2005).

Por otro lado, está el Comité de Administración y el Administrador. Este último tiene entre sus funciones las siguientes:

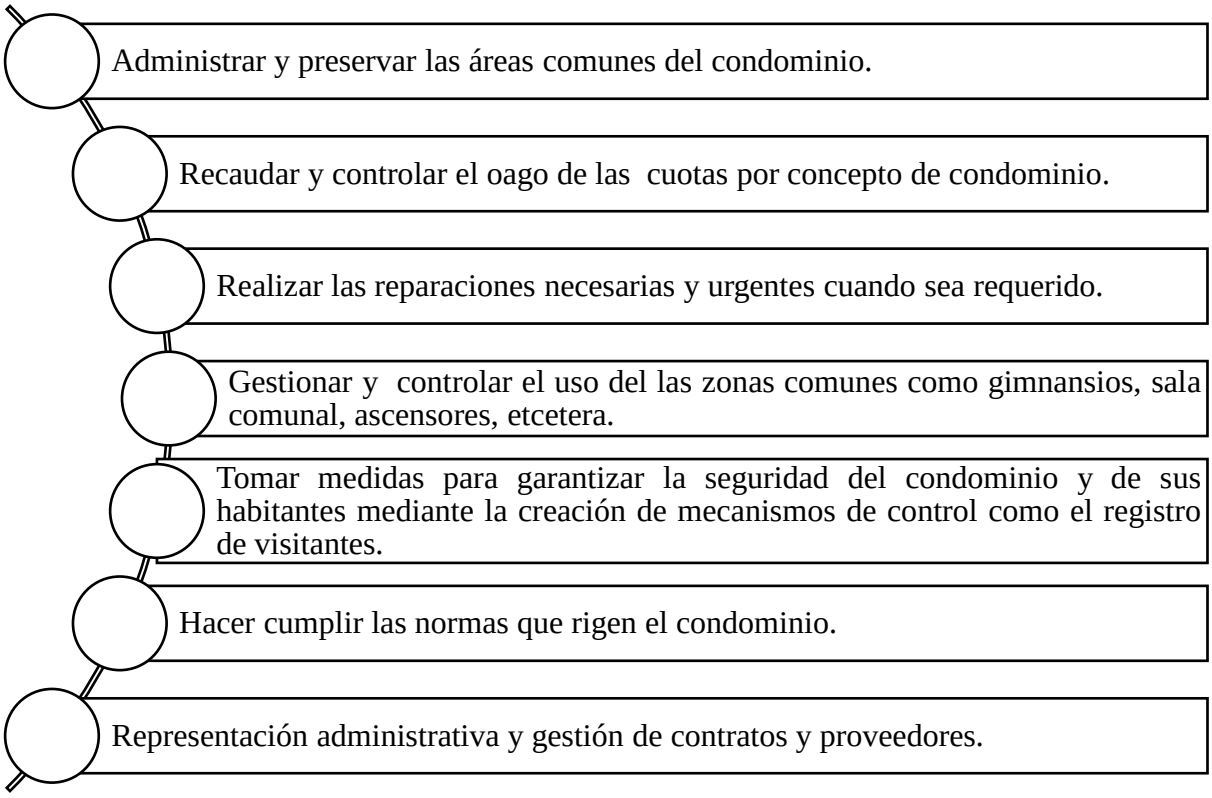


Figura 3.Funciones del administrador del condominio.
Fuente: (Ley de Propiedad Horizontal, 2005)

Estos órganos en conjunto tienen responsabilidades y roles en todos los ámbitos del condominio. Ellos trabajan en colaboración para asegurar una gestión eficiente, una convivencia armoniosa y el cumplimiento de las normativas legales y reglamentos internos en el condominio. Cada uno desempeña un papel importante en el funcionamiento integral de la comunidad de propietarios.

5.1.4. Principales problemáticas en la administración de condominios

La vida en un condominio puede presentar diversas problemáticas que afectan la convivencia y el buen funcionamiento de este. Entre ellas se encuentran los conflictos entre copropietarios (Gracida, 2022). Estos representan uno de los principales desafíos en la gestión de condominios. Estos desacuerdos pueden surgir por decisiones importantes, el uso de áreas

comunes, el pago de cuotas o el comportamiento de los residentes, lo que genera tensiones y afecta la convivencia en el edificio.

Igualmente, se manifiesta el problema del mantenimiento deficiente de las áreas comunes, como jardines, piscinas o infraestructuras, es otro problema frecuente que afecta la calidad de vida de los residentes y disminuye el valor de las propiedades. Además, están las dificultades relacionadas con la gestión financiera específicamente para administrar los recursos financieros del condominio, incluyendo la recaudación de cuotas, la elaboración de presupuestos y el manejo transparente de los fondos.

Por otra parte, la falta de participación de algunos copropietarios en las actividades y decisiones del condominio complica la toma de decisiones y el logro de objetivos comunes. Unido a ello, el cumplimiento legal, ya que el desconocimiento o incumplimiento de normativas legales y reglamentos internos puede generar problemas legales y sanciones que afectan la reputación y la estabilidad del condominio.

También, se deben mencionar, los problemas relacionados con la seguridad, como la falta de control de acceso, deficiencias en sistemas de vigilancia o medidas de prevención insuficientes, representan un riesgo para la seguridad de los residentes y sus propiedades. Finalmente, una comunicación ineficiente entre la administración, la Junta Directiva y los copropietarios puede generar malentendidos, falta de información y dificultades para resolver problemas de manera efectiva, contribuyendo a la complejidad de la gestión del condominio.

5.1.5. Sistemas de control de condominio. Importancia

Para revisar el tema expuesto se debe comenzar por decir que, un sistema de control es una estructura organizada de dispositivos, procesos y protocolos diseñados para supervisar, regular y dirigir el funcionamiento de una organización, en este caso de un condominio (Alfaro, 2018). Estos son fundamentales para mantener un nivel de calidad, eficiencia, seguridad y cumplimiento de objetivos. Además, para simplificar la gestión y administración de la propiedad.

En un condominio, los sistemas de control desempeñan un papel fundamental en la gestión eficiente y segura de diversas áreas. Uno de los sistemas más importantes es el control de acceso, que utiliza tecnologías como tarjetas, códigos o lectores biométricos para regular quién puede ingresar al condominio y a qué áreas específicas. Esto no solo garantiza la seguridad de los residentes y sus propiedades, sino que también ayuda a controlar el acceso a zonas restringidas, como áreas de servicios o instalaciones comunes de valor.

Se debe plantear que, otro aspecto fundamental en el condominio es la vigilancia a través de sistemas de cámaras de seguridad, que permiten monitorear y registrar actividades en áreas comunes y exteriores. Estos sistemas contribuyen significativamente a la prevención de delitos, la identificación de situaciones de riesgo y la protección general del condominio. Además, el control de iluminación y climatización automatizado optimiza el consumo energético y asegura el confort de los residentes al regular de manera eficiente la iluminación y temperatura en áreas comunes.

En ese sentido, los sistemas de gestión de visitantes son primordiales, ya que facilitan el registro y control de las personas que ingresan al condominio, asegurando que los visitantes

estén autorizados por los residentes o la administración. Esto ayuda a mantener un ambiente seguro y controlado, evitando intrusiones no deseadas o situaciones de riesgo.

Además, los sistemas de administración y contabilidad son fundamentales para la gestión financiera del condominio. Estos incluyen software especializado que facilita la recaudación de cuotas, la elaboración de presupuestos, el pago a proveedores y la generación de informes financieros, asegurando una administración transparente y eficiente de los recursos económicos del condominio.

Por último, los sistemas de mantenimiento preventivo permiten prolongar la vida útil de las infraestructuras y equipos del condominio. Estos programas y protocolos garantizan un mantenimiento regular y adecuado de las instalaciones comunes, reduciendo la probabilidad de averías y asegurando un ambiente seguro, cómodo y bien cuidado para todos los residentes. En conjunto, estos sistemas de control contribuyen a crear un entorno condominal que brinda seguridad, comodidad y calidad de vida a sus habitantes.

De manera general, la importancia de los sistemas de control radica en su capacidad para establecer y mantener estándares de desempeño, asegurando que los procesos se desarrollen de manera ordenada y eficaz. Permiten detectar y corregir desviaciones o anomalías en tiempo real, lo que contribuye a prevenir errores, minimizar riesgos y optimizar el uso de recursos. Además, proporcionan información valiosa para la toma de decisiones estratégicas, al ofrecer datos precisos y actualizados sobre el rendimiento y funcionamiento adecuado del condominio.

5.1.6. Aplicación de la tecnología en la administración de condominio

Para revisar la aplicación de la tecnología en un condominio se debe partir con la definición de software que es un grupo de programas y datos que están almacenados en una computadora. Este constituye la componente lógica que facilita que los dispositivos puedan ser utilizados (Vizcaíno y otros, 2021).

En ese orden, el software se refiere al conjunto de elementos lógicos e intangibles de un sistema informático, que incluyen los programas de computadora, procedimientos, reglas, documentación y datos necesarios para llevar a cabo funciones específicas. Es la parte lógica que permite la ejecución de tareas y operaciones dentro de un sistema de computación, conformando un elemento fundamental en el funcionamiento de cualquier dispositivo informático (Pantaleo & Rinaudo, 2017).

Ahora bien, específicamente, el Software de administración de condominios es una herramienta fundamental en la gestión eficiente y efectiva de edificios residenciales o complejos habitacionales. Este ofrece una amplia gama de funcionalidades diseñadas para optimizar la administración y mejorar la experiencia tanto de los propietarios como del personal encargado de la gestión del condominio.

Por otro lado, un software de administración de edificios es un tipo de programa informático diseñado para asistir en la gestión y supervisión de diversos aspectos relacionados con la operación de edificios. Estas herramientas abarcan desde el control de seguridad hasta el mantenimiento, recursos y operaciones generales del edificio. Entre las funcionalidades que suelen ofrecer se encuentran la programación y seguimiento de mantenimiento preventivo y correctivo, gestión de activos e inventario, control de costos, así como la generación de informes de seguridad y cumplimiento (Coronado, 2019).

Adicionalmente, algunos softwares de administración de edificios pueden incluir características avanzadas como la supervisión remota de dispositivos y sistemas de seguridad, junto con la integración de sistemas de automatización para regular el consumo de energía y mejorar la eficiencia energética del edificio. La aplicación de la tecnología facilita el funcionamiento del condominio.

Su puesta en práctica abarca múltiples aspectos que mejoran la calidad de vida, la eficiencia operativa y la seguridad de los residentes. En primer lugar, la tecnología facilita la gestión administrativa y financiera del condominio a través de sistemas de software especializados que automatizan tareas como la recaudación de cuotas, la elaboración de presupuestos y la generación de informes financieros, lo que garantiza una administración transparente y eficiente de los recursos. La aplicación de un software en un condominio residencial puede tener varias ventajas, entre las razones más comunes que evidencian su valor están:

Aspectos que se benefician con la aplicación de un sofware en el contexto de un condominio.	**Eficiencia en la gestión:** El uso de software especializado permite automatizar muchas tareas administrativas, como la gestión de pagos de cuotas, la programación de mantenimientos, la comunicación con los residentes y el seguimiento de incidencias. Esto ahorra tiempo y recursos para la administración del condominio.
	Mayor transparencia: Los sistemas de software pueden proporcionar a los residentes acceso a información transparente sobre las finanzas del condominio, el estado de las áreas comunes, los servicios disponibles y las decisiones tomadas por la administración. Esto fomenta la confianza y la participación de los residentes en la vida comunitaria.
	Mejora en la comunicación: Las plataformas de software facilitan la comunicación fluida entre la administración y los residentes, así como entre los propios residentes. Se pueden enviar notificaciones, convocatorias de reuniones, encuestas y compartir información relevante de manera rápida y eficiente.
	Optimización de recursos: Con herramientas de software, es posible optimizar el uso de recursos como la energía, el agua y el mantenimiento de infraestructuras. Se pueden implementar sistemas de gestión inteligente que reduzcan el consumo y los costos operativos del condominio.
	Mejora en la seguridad: La tecnología puede integrarse con sistemas de seguridad como cámaras de vigilancia, control de acceso mediante tarjetas o códigos, y alarmas conectadas a centros de monitoreo. Esto aumenta la seguridad de los residentes y sus propiedades.
	Experiencia del residente: Al ofrecer servicios digitales como reservas en áreas comunes, reporte de averías o seguimiento de solicitudes, se mejora la experiencia del residente, haciéndola más cómoda y satisfactoria.

Figura 4. Ventajas que representa la aplicación de un software en el contexto de la administración de un condominio.
Fuente: (Alfaro, 2018)

Se debe apuntar que, la tecnología en la gestión de condominio busca que los propietarios puedan operar de manera autónoma, especialmente al momento de recibir avisos y aplicar sanciones. De manera conveniente, estas plataformas notifican al usuario sobre los plazos para abonar sus gastos habituales o cuando se encuentra en situación de morosidad. En resumen, aplicar tecnología mediante software en un condominio residencial puede transformar la

gestión, la comunicación y la experiencia de los residentes, aumentando la eficiencia, la transparencia y la calidad de vida en la comunidad.

5.1.7. Tipos de software de administración de condominio disponibles.

Sus ventajas

En el mercado actualmente existen varios tipos de software cuyo fin es aplicarlos en la administración de condominios entre ellos se encuentran los siguientes:

- Software para la gestión financiera: permite llevar un registro detallado de los ingresos y gastos del condominio, incluyendo la recaudación de cuotas, pagos de servicios, mantenimiento de áreas comunes y presupuestos.
- Software de comunicación: facilita la comunicación interna y externa del condominio, permitiendo enviar notificaciones, convocatorias, circulares y mantener un registro de las comunicaciones realizadas.
- Software para la reserva de eventos y áreas comunes: permite gestionar reservas de espacios comunes como salones de eventos, áreas de recreación o piscinas, así como organizar eventos y actividades para los residentes.
- Software para mantenimiento: este contribuye a tramitar las solicitudes de mantenimiento, al igual que a asignar tareas a proveedores o personal interno, y realizar seguimiento de los trabajos realizados.
- Software para la seguridad: estos integran funcionalidades de control de acceso, monitoreo de cámaras de seguridad y registro de incidentes para garantizar la seguridad de los residentes.
- Software de documentos: permite almacenar y gestionar documentos importantes del condominio, como reglamentos internos, contratos, actas de asamblea, entre otros.

Como se aprecia existen diferentes clases de software aplicables a la administración de condominio. Estos por su funcionalidad constituyen una herramienta completa que facilita la gestión integral del inmueble, mejorando la eficiencia, la calidad de vida de sus residentes y el funcionamiento de la administración.

5.2. Desarrollo web

El desarrollo web abarca todo el proceso de construcción y mantenimiento de sitios web y aplicaciones web. En este proceso, se integran diferentes disciplinas y técnicas para crear plataformas funcionales y atractivas para los usuarios (Rubiales, 2021). Específicamente el diseño, se encarga de la apariencia visual y la estructura del sitio, asegurando una buena experiencia de usuario y una interfaz intuitiva. Esto implica trabajar con elementos gráficos, colores, tipografías y adaptar el diseño para que sea compatible con diferentes dispositivos, como computadoras, tablets y smartphones.

Asimismo, el desarrollo web se conforma por varias actividades que van desde el diseño hasta la implementación de funcionalidades. En el diseño web, se crea la apariencia visual del sitio, definiendo la disposición de elementos, colores, tipografías y estilos. Esto se complementa con el desarrollo front-end, donde se utiliza HTML, CSS y JavaScript para dar vida a la interfaz de usuario, permitiendo la interacción y navegación fluida de los usuarios en el sitio (Mattos, 2021).

Por otro lado, en cuanto a la programación, que es el aspecto técnico donde se utilizan diversos lenguajes como HTML, CSS, JavaScript, PHP, entre otros, para desarrollar la funcionalidad del sitio. Esto incluye desde la creación de páginas estáticas hasta sistemas complejos de gestión de contenidos o aplicaciones interactivas. Además, se emplean bases de

datos para almacenar y gestionar información, como perfiles de usuarios, productos o contenido dinámico, utilizando sistemas como MySQL, PostgreSQL, entre otros (Gómez, 2023).

Frontend

Corresponde en esta parte referirse a aspectos fundamentales del desarrollo como el *frontend*. Este se refiere a la parte de una aplicación o sistema que interactúa directamente con los usuarios finales. Es la interfaz a través de la cual los usuarios interactúan con el software. Este se encarga de la presentación y la interacción del usuario, incluidos el diseño, la disposición, los estilos y la funcionalidad de la interfaz de usuario (Gómez, 2023).

Se debe exponer que, el *frontend* trabaja en conjunto con el *backend*, que es la parte de la aplicación que se encarga del procesamiento de datos, la lógica de negocio y la comunicación con bases de datos u otros servicios externos. Ellos de manera unida forman una aplicación completa que ofrece una experiencia de usuario completa.

Backend

Por su lado, el desarrollo back-end se enfoca en la parte interna del sitio, gestionando bases de datos, lógica de negocio y comunicación con el servidor. Aquí se utilizan lenguajes como PHP, Python, Ruby, Java, entre otros, junto con sistemas de gestión de bases de datos para almacenar y gestionar información de manera eficiente (Hernández & Baquero, 2021).

En ese orden, el *backend* es la parte de una aplicación o sistema informático que se encarga de procesar los datos. En él se realiza la lógica de negocio y se gestiona la comunicación con la base de datos y otros servicios externos. Es la capa que está detrás de la interfaz de usuario y

que se encarga de la funcionalidad que no es visible para el usuario final (Gómez, 2023). Al respecto, es necesario detenerse en los siguientes aspectos:

- Gestión de datos: es el almacenamiento, recuperación, actualización y eliminación de datos en la base de datos (Celí, Boné, & Mora, 2023).
- Lógica de negocio: se refiere a la implementación de reglas de negocio, algoritmos y procesos necesarios para que la aplicación funcione correctamente (Celí, Boné, & Mora, 2023).
- Seguridad: consiste en poner en práctica las medidas de seguridad para proteger los datos y prevenir accesos no autorizados (Celí, Boné, & Mora, 2023).
- Autenticación y autorización: Esto se dirige a la gestión de la identidad de los usuarios, autenticación de usuarios y autorización de acceso a recursos y funcionalidades (Celí, Boné, & Mora, 2023).
- Integración con servicios externos: se enfoca en la comunicación con otros sistemas o servicios externos, como APIs de terceros, servicios web, sistemas de pago, etc (Celí, Boné, & Mora, 2023).

5.2.1. Lenguajes de programación para desarrollo web (HTML, CSS, JavaScript).

Los lenguajes de programación son conjuntos de instrucciones y reglas que permiten a los desarrolladores comunicarse con las computadoras y crear software. Cada lenguaje tiene su propia sintaxis, semántica y conjunto de reglas que determinan cómo se escriben y organizan las instrucciones (Sestoft, 2022).

Los lenguajes de programación se utilizan para escribir programas y aplicaciones de software que pueden realizar una variedad de tareas, desde simples cálculos hasta aplicaciones empresariales complejas y sistemas operativos. Algunos lenguajes están diseñados para

propósitos específicos, mientras que otros son más generales y versátiles. Algunos ejemplos de lenguajes de programación populares incluyen: Java, Python, JavaScript, C#, C++, PHP, Swift, Ruby.

Por su lado, **HTML** (*HyperText Markup Language*) es el lenguaje fundamental para la creación de páginas web. Su función principal es definir la estructura y contenido de una página web mediante el uso de etiquetas. Estas etiquetas proporcionan información al navegador sobre cómo debe mostrar el contenido, como encabezados, párrafos, enlaces, imágenes y formularios. HTML no es un lenguaje de programación en sí mismo, sino un lenguaje de marcado que se enfoca en la presentación y organización de la información en una página web (Gerginov, 2020).

Por otra parte, **CSS** (*Cascading Style Sheets*) se encarga de definir el aspecto visual de una página web creada con HTML. Permite aplicar estilos y diseños a los elementos HTML, como colores, fuentes, márgenes, alineaciones y efectos visuales. CSS utiliza reglas de estilo que especifican cómo deben lucir los elementos HTML en la página. Esta separación entre HTML y CSS facilita la modificación y mantenimiento del diseño de una página web sin afectar su estructura (Gerginov, 2020).

Finalmente, **JavaScript** es un lenguaje de programación de alto nivel que se utiliza para agregar interactividad y funcionalidad dinámica a las páginas web. A diferencia de HTML y CSS, JavaScript es un lenguaje de programación completo que permite realizar acciones como validar formularios, manipular elementos del DOM (*Document Object Model*), crear animaciones, comunicarse con servidores para cargar contenido dinámicamente, entre otras cosas. JavaScript es esencial para crear experiencias web interactivas y dinámicas. En conjunto,

HTML, CSS y JavaScript forman el trío fundamental para el desarrollo web moderno, permitiendo crear páginas web atractivas, funcionales y con una gran experiencia de usuario (Rubiales, 2021).

Se deben exponer los lenguajes de programación que mayor uso tienen actualmente entre ellos se destacan: java, Python, JavaScript, C#, C++, PHP, Swift, Ruby. En ese sentido, es necesario detenerse en C# por ser el que se aplica en esta investigación:

C#

En relación con el lenguaje C# es importante iniciar por decir que es desarrollado por Microsoft como parte de su plataforma .NET. Este fue creado por *Anders Hejlsberg* y su equipo durante la década de 1990 y se lanzó en 2000 como parte de la plataforma *.NET Framework*. Desde entonces, ha experimentado varias actualizaciones y mejoras, y sigue siendo uno de los lenguajes de programación más populares en la actualidad (Arrioja, 2018).

Cabe mencionar, que C# es un lenguaje de programación de propósito general que está diseñado para ser simple, moderno, seguro y orientado a objetos. Está influenciado por varios otros lenguajes, incluidos C++, Java y Delphi (Arrioja, 2018). Algunas de las características fundamentales de C# incluyen:

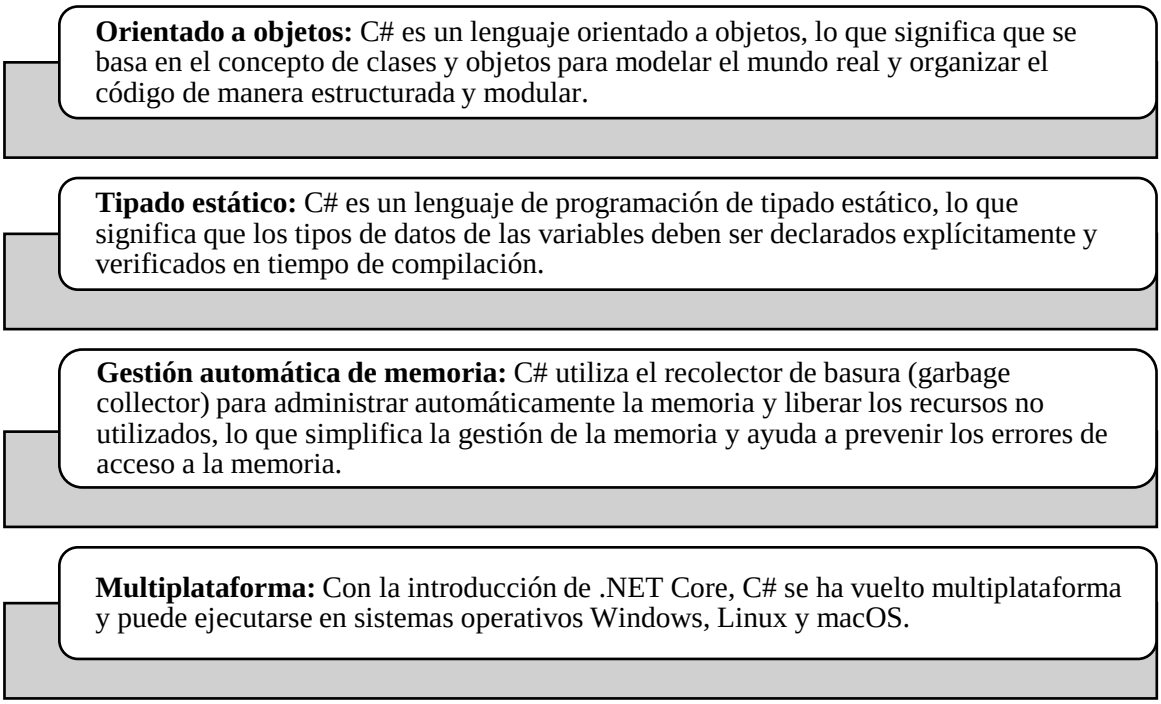


Figura 5.Características del lenguaje de programación C#.
Fuente: (Arrioja, 2018)**.**

Como se aprecia, el lenguaje estudiado, es rico en características porque incluye soporte para programación orientada a eventos, programación asincrónica, LINQ (*Language Integrated Query*), manejo de excepciones, propiedades, indexadores, delegados, eventos, entre otros.

Igualmente, C# se utiliza ampliamente en una variedad de escenarios de desarrollo de software, incluyendo desarrollo de aplicaciones de escritorio, desarrollo web, desarrollo de aplicaciones móviles, desarrollo de juegos, desarrollo de servicios web, entre otros. Además, C# es el lenguaje principal utilizado en el desarrollo de aplicaciones en la plataforma *.NET* de Microsoft.

5.3. Framework

Se debe estudiar lo referente al *framework* que es un conjunto de herramientas, bibliotecas y reglas predefinidas que proporcionan una estructura para el desarrollo de software. Estas

permiten a los desarrolladores construir aplicaciones de manera más eficiente al proporcionar una base sólida y predefinida sobre la cual programar (Unir, 2022).

En esa línea, los *frameworks* suelen incluir componentes como bibliotecas de funciones, módulos de código, patrones de diseño, estructuras de directorios y reglas de convención sobre configuración. Estos componentes ayudan a los desarrolladores a escribir código más limpio, estructurado y mantenible al proporcionar soluciones predefinidas para tareas comunes y problemas recurrentes (Unir, 2022).

Por otro lado, estas tecnologías pueden ser específicas para un lenguaje de programación en particular o pueden ser más generales y utilizables en varios de ellos. Algunas están diseñadas para propósitos específicos, como desarrollo web, aplicaciones móviles, desarrollo de juegos, aprendizaje automático, etcétera. Entre los más utilizados se encuentran: *Django, Ruby on Rails, AngularJS / Angular, React, Spring Framework, Express.js.* A continuación, se amplía en relación con las tecnologías más utilizadas en esta investigación.

5.3.1. ASP.NET.Core

Se debe exponer que, el *ASP.NET Core* es un *framework* de desarrollo de software de código abierto para la construcción de aplicaciones web modernas y escalables. Este constituye una evolución de *ASP.NET*, pero está diseñado para ser más modular, flexible, rápido y multiplataforma. Algunas características y ventajas clave se detallan a continuación:

- Multiplataforma: ASP.NET Core es compatible con Windows, Linux y macOS, lo que permite desarrollar y ejecutar aplicaciones en una variedad de sistemas operativos (Barbettini, 2018).

- Código abierto: ASP.NET Core es de código abierto y está disponible en GitHub, lo que permite a la comunidad contribuir, mejorar y extender el marco de trabajo (Barbettini, 2018).
- Rendimiento mejorado: ASP.NET Core está diseñado para ser más rápido y eficiente que las versiones anteriores de ASP.NET. Utiliza tecnologías como Kestrel (un servidor web ligero), la compilación anticipada (precompilación) y la inyección de dependencias integrada para mejorar el rendimiento de las aplicaciones (Barbettini, 2018).
- Modularidad y flexibilidad: ASP.NET Core es altamente modular, lo que significa que puedes elegir y utilizar solo los componentes necesarios para tu aplicación. También es compatible con diferentes modelos de desarrollo, como MVC (Model-View-Controller), API RESTful, SignalR (para aplicaciones en tiempo real) y Razor Pages, entre otros (Barbettini, 2018).
- ASP.NET Core MVC: Es un marco de trabajo ligero y potente para construir aplicaciones web basadas en el patrón de diseño Modelo-Vista-Controlador (MVC) (Barbettini, 2018).
- ASP.NET Core Web API: Permite la creación de servicios web RESTful utilizando el enfoque basado en controladores para la definición de rutas y la lógica de manejo de solicitudes (Barbettini, 2018).
- Seguridad integrada: ASP.NET Core incluye características integradas de seguridad, como autenticación y autorización basadas en identidad, y soporte para la protección contra ataques comunes, como la inyección de SQL y los ataques de scripting entre sitios (XSS) (Barbettini, 2018).

5.3.2. Entity Framework

Entity Framework es un marco de trabajo de mapeo objeto-relacional (ORM) para el desarrollo de aplicaciones de bases de datos en la plataforma .NET. Este permite a los desarrolladores trabajar con datos en forma de objetos y clases en lugar de escribir consultas

SQL directamente contra la base de datos. Además, se encarga de traducir las operaciones de bases de datos en consultas SQL y administrar la comunicación entre la aplicación y la base de datos (Muñoz, 2019). Algunas de sus características principales son:

- Mapeo de entidades: Permite definir clases que representan entidades en la base de datos y mapearlas a tablas y relaciones en la base de datos.
- LINQ to Entities: Permite consultar datos utilizando LINQ (Language Integrated Query) directamente contra las entidades de la base de datos, lo que proporciona una sintaxis de consulta tipo SQL en código C# o VB.NET.
- Operaciones CRUD: Entity Framework facilita la creación, lectura, actualización y eliminación de datos en la base de datos utilizando operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en las entidades.
- Control de versiones y concurrencia: Proporciona soporte para el control de versiones de datos y la resolución de conflictos de concurrencia automáticamente.
- Soporte para diferentes bases de datos: Entity Framework es compatible con una variedad de bases de datos relacionales, incluyendo SQL Server, MySQL, PostgreSQL, SQLite y Oracle, entre otras.

En resumen, Entity Framework simplifica el desarrollo de aplicaciones que interactúan con bases de datos al proporcionar una capa de abstracción entre la aplicación y la base de datos subyacente, lo que facilita la gestión y manipulación de datos a través de objetos y consultas LINQ en el código .NET.

5.4. Hosting

El *hosting* se refiere al servicio de alojamiento de sitios web o aplicaciones en servidores remotos que están disponibles en Internet (Martínez, 2021). En pocas palabras, el hosting

permite que tu sitio web o aplicación sea accesible en línea para los usuarios de Internet en todo el mundo.

5.4.1. Tipos de *hosting*

Asimismo, el hosting puede ser de diferentes tipos, los que se exponen a continuación:

- **Hosting compartido:** En este tipo de hosting, varios sitios web comparten los recursos de un solo servidor. Es una opción económica y adecuada para sitios web pequeños y de bajo tráfico.
- **Hosting VPS (Servidor Privado Virtual):** En un entorno de hosting VPS, un servidor físico se divide en múltiples servidores virtuales independientes que funcionan de manera aislada. Esto ofrece más control y recursos dedicados en comparación con el hosting compartido.
- **Hosting dedicado:** Con el hosting dedicado, obtienes un servidor completo para ti solo. Tienes control total sobre el servidor y sus recursos, lo que lo hace ideal para sitios web grandes y de alto tráfico que requieren un rendimiento y una personalización máximos.
- **Hosting en la nube:** En el hosting en la nube, tu sitio web o aplicación se ejecuta en una red de servidores virtuales distribuidos en la nube. Ofrece escalabilidad, flexibilidad y rendimiento mejorados, ya que puedes escalar tus recursos de forma dinámica según sea necesario.
- **Hosting administrado:** Con el hosting administrado, el proveedor de hosting se encarga de administrar y mantener el servidor, incluida la configuración, la seguridad y las actualizaciones del sistema. Esto libera al propietario del sitio web de tareas técnicas y de mantenimiento.

Se debe apuntar que, la elección del tipo de *hosting* depende de varios factores, como: el tamaño y el tráfico del sitio web, los recursos requeridos, el presupuesto y las necesidades específicas del proyecto. Por ello se debe investigar y comparar diferentes opciones de hosting para encontrar la mejor solución para tus necesidades.

5.4.2. Servicios de hosting en la nube

Los servicios de hosting en la nube son plataformas que ofrecen recursos informáticos, como servidores, almacenamiento, bases de datos y software, a través de Internet. Estos recursos están alojados en centros de datos distribuidos en todo el mundo y se accede a ellos a través de una conexión a Internet (Martínez, 2021). Entre las ventajas de esta clase *hosting* están:

- **Escalabilidad**: Esta dada en su capacidad para escalar recursos de manera rápida y eficiente. Esto significa que puedes aumentar o disminuir la capacidad de tu infraestructura según las necesidades del negocio, sin la necesidad de adquirir hardware adicional o realizar cambios físicos en tu entorno (Martínez, 2021).
- **Flexibilidad**: Este ofrece una amplia gama de opciones de configuración y personalización para adaptarse a las necesidades específicas de cada negocio. Puedes elegir entre diferentes tipos de instancias de servidor, sistemas operativos, bases de datos, servicios de almacenamiento y más, según tus requerimientos (Martínez, 2021).
- **Disponibilidad y fiabilidad**: Los proveedores de servicios en la nube suelen utilizar infraestructuras redundantes y distribuidas geográficamente para garantizar una alta disponibilidad y fiabilidad de los servicios. Esto significa que tus aplicaciones y datos estarán protegidos contra fallas de hardware, errores humanos y otros eventos adversos (Martínez, 2021).

- **Pago por uso**: Muchos servicios de hosting en la nube utilizan un modelo de precios basado en el uso, lo que significa que solo pagas por los recursos que consumes. Esto puede resultar más económico y eficiente que los modelos de precios tradicionales basados en planes fijos, ya que te permite ajustar tus costos según la demanda y evitar gastos innecesarios (Martínez, 2021).

Por otra parte, entre los proveedores más conocidos de servicios de hosting en la nube incluyen:

- **Amazon Web Services (AWS):** Es uno de los proveedores líderes en el mercado de servicios en la nube, ofreciendo una amplia gama de servicios y soluciones para empresas de todos los tamaños.
- **Microsoft Azure:** Es la plataforma de servicios en la nube de Microsoft, que ofrece una amplia variedad de servicios, desde almacenamiento y computación hasta inteligencia artificial y análisis de datos.
- **Google Cloud Platform (GCP):** Es la plataforma de servicios en la nube de Google, que proporciona recursos informáticos escalables y servicios gestionados para el desarrollo, la implementación y la administración de aplicaciones y sitios web.
- **IBM Cloud:** Es la plataforma de servicios en la nube de IBM, que ofrece servicios de infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS) para empresas de todo el mundo.

5.5. Bases de datos

Las bases de datos son sistemas fundamentales diseñados para almacenar, gestionar y recuperar información de manera eficiente. Son como almacenes organizados de datos que se estructuran de forma específica para facilitar su uso y administración. Permiten a los usuarios

realizar una variedad de operaciones como agregar, modificar, eliminar y buscar datos, y se encuentran en una amplia gama de aplicaciones, desde simples sistemas de almacenamiento hasta complejos sistemas de gestión empresarial (Silberschatz y otros, 2019).

Asimismo, una de las definiciones más importantes de las bases de datos son las tablas, que son la estructura fundamental en las bases de datos relacionales. Estas tablas representan conjuntos de datos organizados en filas y columnas, donde cada fila es un registro que representa una entidad y cada columna es un atributo de esa entidad. Cada registro contiene datos relacionados y representa una instancia específica de una entidad en la base de datos. Los campos, por su parte, son las columnas de una tabla y representan atributos específicos de los registros, almacenando valores individuales de datos (Silberschatz, Korth, & Sudarshan, 2019).

Por otro lado, se deben resaltar las claves primarias que constituyen campos o conjuntos de campos que identifican de manera única cada registro en una tabla. Estas claves garantizan la integridad de los datos y se utilizan como referencia para relacionar tablas entre sí. Las consultas son comandos utilizados para recuperar, modificar o eliminar datos de una base de datos, permitiendo a los usuarios realizar operaciones específicas en los datos almacenados (Vicente & Vivas, 2014).

Además, las bases de datos suelen contar con sistemas de gestión de usuarios y permisos que controlan quién puede acceder a qué datos y qué operaciones pueden realizar. Esto asegura la seguridad y la integridad de la información almacenada en la base de datos, permitiendo un uso controlado y eficiente de los datos (Mora, 2014). Dichas bases de datos son sistemas diseñados para almacenar, gestionar y recuperar información

5.5.1. Tipos de bases de datos

Existen diferentes tipos de bases de datos, como las relacionales (SQL), las NoSQL, las de series temporales, las de documentos, entre otras, cada una con características y aplicaciones específicas.

5.5.1.1. SQL

SQL es el lenguaje estándar ANSI/ISO utilizado para definir, manipular y controlar bases de datos relacionales. Se trata de un lenguaje declarativo, lo que significa que solo es necesario especificar lo que se desea realizar (Escofet, 2019).

Al respecto, el SQL, o Lenguaje de Consulta Estructurada, facilita la gestión de datos almacenados en un sistema de base de datos relacional. Este lenguaje permite a los usuarios definir qué datos necesitan sin detallar el procedimiento para obtener esos datos. Esto hace que las operaciones con bases de datos sean más intuitivas y accesibles para usuarios no especializados en programación (Escofet, 2019).

Además de la función principal de consulta de datos, SQL permite a los usuarios definir la estructura de las bases de datos, modificar datos y establecer permisos de acceso. Esto lo convierte en una herramienta importante para los administradores de bases de datos y desarrolladores que necesitan gestionar eficientemente grandes cantidades de información y asegurar su integridad y seguridad (Escofet, 2019).

5.5.1.2. NoSQL

NoSQL, significa "No solo SQL" (Not Only SQL), se refiere a una familia de sistemas de gestión de bases de datos que difieren del modelo tradicional de bases de datos relacionales SQL. Estos sistemas están diseñados para manejar grandes volúmenes de datos distribuidos y

para proporcionar alta escalabilidad y flexibilidad, siendo una solución efectiva para aplicaciones que manejan tipos variados de datos y que requieren escalabilidad horizontal (Gracia del Busto y Yanes, 2012).

Cabe mencionar que una característica distintiva de las bases de datos NoSQL es su capacidad para operar con esquemas flexibles. A diferencia de las bases de datos relacionales, que requieren un esquema definido y estricto, las bases de datos NoSQL permiten un esquema dinámico y más adaptable. Esto es especialmente útil para manejar datos no estructurados o semi-estructurados como documentos, mensajes de texto, y contenido multimedia, facilitando su integración y gestión en aplicaciones modernas (Gracia del Busto y Yanes, 2012).

Además, las bases de datos NoSQL son conocidas por su escalabilidad horizontal, que permite expandir la capacidad de almacenamiento y procesamiento añadiendo más nodos al sistema, en lugar de depender de un solo servidor con recursos limitados. Esta característica las hace ideales para aplicaciones web a gran escala, análisis de big data y aplicaciones que requieren un alto rendimiento en tiempo real (Gracia del Busto y Yanes, 2012).

Entre los sistemas que se emplean para gestionar dichas bases se encuentran los que se explican a continuación:

5.5.2. Sistemas de gestión de bases de datos (SGBD)

Los Sistemas de Gestión de Bases de Datos (SGBD) son software especializado diseñado para gestionar, almacenar, manipular y recuperar datos de manera eficiente y segura en bases de datos. Estos sistemas proporcionan una interfaz entre los usuarios y la base de datos,

permitiendo a los usuarios realizar consultas y realizar operaciones en los datos almacenados (Mora, 2014). Estos ofrecen una variedad de características y funcionalidades, incluyendo:

- Modelado de datos: Los SGBD permiten definir la estructura de los datos utilizando modelos de datos como el modelo relacional, el modelo de red, el modelo jerárquico, entre otros.
- Creación y gestión de bases de datos: Permiten crear y administrar bases de datos, incluyendo la creación de tablas, la definición de relaciones entre tablas, la gestión de índices y la configuración de la seguridad de la base de datos.
- Consultas: Los SGBD permiten a los usuarios realizar consultas para recuperar datos de la base de datos utilizando lenguajes como SQL (Structured Query Language).
- Integridad y consistencia de los datos: Los SGBD garantizan la integridad y la consistencia de los datos mediante la aplicación de restricciones de integridad, como las claves primarias, las claves foráneas y las restricciones de unicidad.
- Seguridad: Proporcionan mecanismos de seguridad para controlar el acceso a los datos y proteger la confidencialidad, la integridad y la disponibilidad de la información.
- Recuperación y copias de seguridad: Los SGBD permiten realizar copias de seguridad y restaurar los datos en caso de pérdida de información o fallos en el sistema.
- Optimización de consultas y rendimiento: Los SGBD incluyen optimizadores de consultas que analizan y ejecutan consultas de manera eficiente para mejorar el rendimiento de las operaciones en la base de datos.

Algunos ejemplos populares de SGBD incluyen: *MySQL*, *PostgreSQL*, *Oracle Database*, *Microsoft SQL Server*, *MongoDB*, entre otros. Estos sistemas se utilizan en una amplia variedad de aplicaciones y entornos, desde pequeñas empresas hasta grandes corporaciones y organizaciones gubernamentales.

5.5.3. SQL Server

SQL Server es un Sistema de Gestión de Bases de Datos Relacionales (SGBD) desarrollado por Microsoft (Silberschatz, Korth, & Sudarshan, 2019). Es una plataforma de base de datos completa que proporciona almacenamiento de datos, administración, análisis, seguridad y capacidades de inteligencia empresarial, entre sus características se destacan:

- Modelo relacional: SQL Server utiliza el modelo relacional para almacenar datos en tablas, que se componen de filas y columnas.
- Lenguaje SQL: Ofrece un potente lenguaje de consulta llamado SQL (Structured Query Language) que permite a los usuarios realizar consultas y manipular datos de manera eficiente.
- Seguridad avanzada: Proporciona mecanismos de seguridad robustos para proteger los datos almacenados en la base de datos, incluyendo autenticación, autorización, cifrado de datos y auditoría.
- Alta disponibilidad y tolerancia a fallos: SQL Server ofrece opciones para configurar la alta disponibilidad y la tolerancia a fallos, incluyendo la replicación de datos, la agrupación de conmutación por error (failover clustering) y la agrupación de bases de datos AlwaysOn.
- Integración con otras tecnologías de Microsoft: Se integra estrechamente con otras tecnologías de Microsoft, como .NET Framework, Visual Studio, Azure y Power BI, lo que facilita el desarrollo y la administración de aplicaciones empresariales.
- Capacidades de análisis y generación de informes: SQL Server incluye herramientas de análisis de datos y generación de informes, como SQL Server Analysis Services (SSAS) y SQL Server Reporting Services (SSRS), que permiten a los usuarios analizar datos y crear informes personalizados.
- Escalabilidad: SQL Server es escalable y puede manejar grandes volúmenes de datos y cargas de trabajo exigentes en entornos empresariales.

Cabe agregar que, SQL Server está disponible en varias ediciones, incluyendo SQL Server Express (gratuito), SQL Server Standard Edition y SQL Server Enterprise Edition. Estos tienen sus propias particularidades y capacidades dirigidas a satisfacer las necesidades de las diferentes organizaciones y aplicaciones (Silberschatz, Korth, & Sudarshan, 2019).

5.5.4. Bases de datos relacionales

Una base de datos relacionales son un tipo de base de datos que organizan los datos en tablas que se relacionan entre sí mediante claves primarias y claves externas. Este modelo de base de datos se basa en los principios de la teoría de conjuntos y proporciona una forma estructurada de almacenar y acceder a la información (Silberschatz, Korth, & Sudarshan, 2019). Se destacan como características de estas las siguientes:

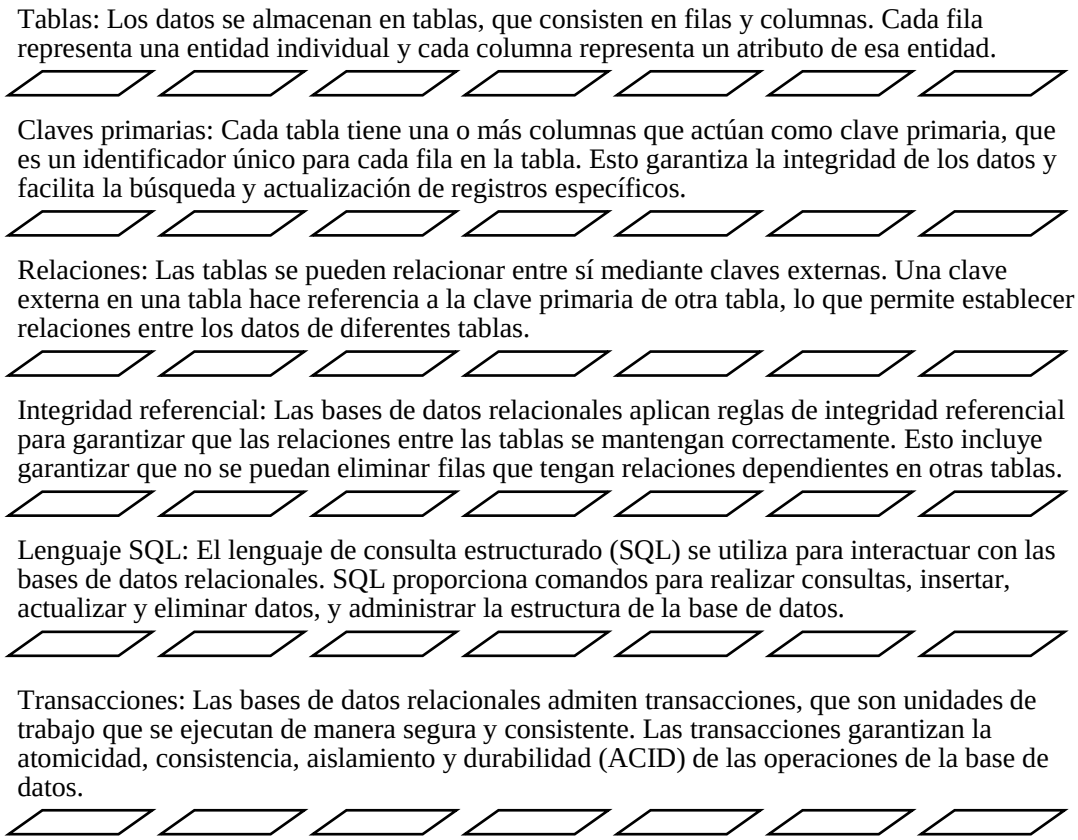


Figura 6. Características de las bases de datos relacionales.
Fuente: (Silberschatz, Korth, & Sudarshan, 2019)

En ese sentido, las bases de datos relacionales son ampliamente utilizadas en una variedad de aplicaciones, desde sistemas de gestión empresarial hasta aplicaciones web y móviles. Ofrecen una estructura flexible, poderosa para almacenar y manipular datos de manera eficiente y segura. Algunos ejemplos populares de sistemas de gestión de bases de datos relacionales (SGBDR) incluyen *MySQL, PostgreSQL, Oracle Database y SQL Server*.

5.6. API

Una API, o Interfaz de Programación de Aplicaciones (*Application Programming Interface*), es un conjunto de reglas y protocolos que permite a diferentes aplicaciones o componentes de software comunicarse entre sí. En pocas palabras, se define cómo interactuar con un sistema de software para lograr ciertas funciones o acceder a determinados recursos (Hernández J. , 2017). Estas pueden proporcionar diferentes tipos de funcionalidades, como:

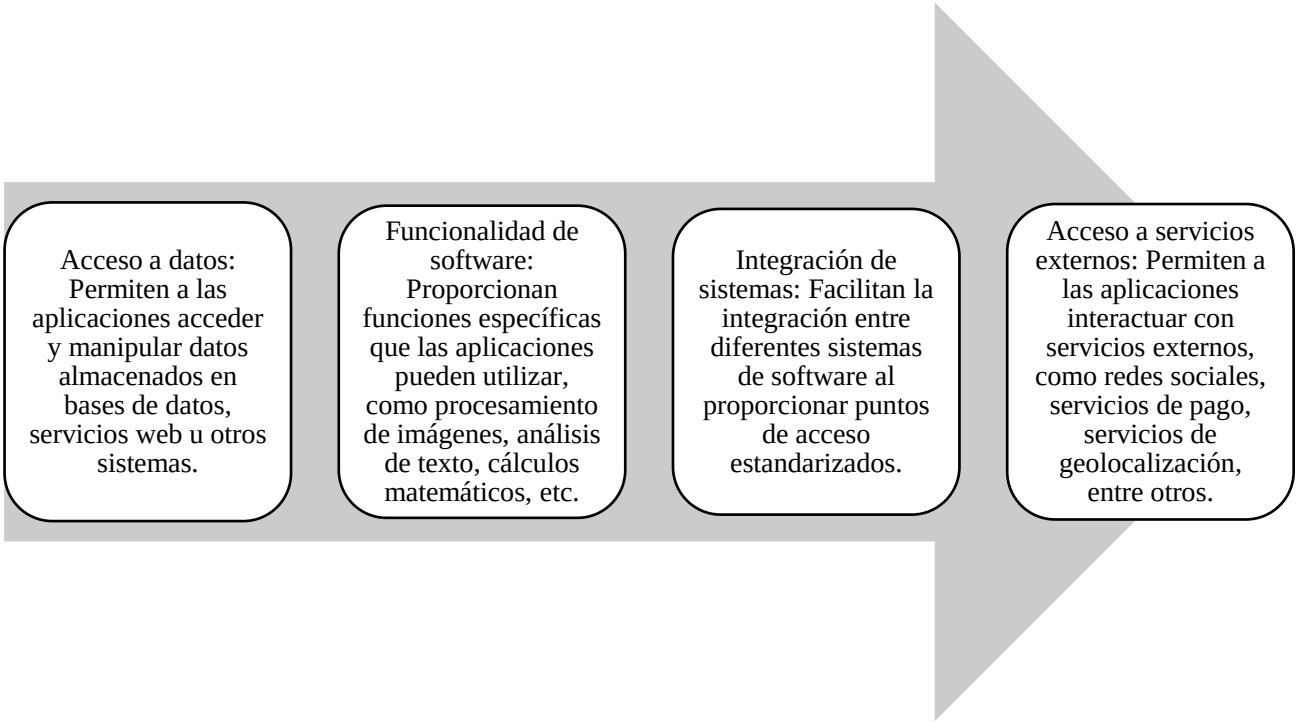


Figura 7.Funcionalidades de la API.
Fuente: *(Hernández J. , 2017)*

Se debe plantear que, las API pueden ser proporcionadas por sistemas operativos, bibliotecas de software, plataformas en la nube y aplicaciones web, entre otros. Utilizan diferentes protocolos de comunicación, como *HTTP, REST, SOAP, GraphQL*, entre otros, para permitir la comunicación entre las aplicaciones cliente y el sistema o servicio que ofrece la API.

5.6.1. Creación de una API

Se debe hacer alusión a la creación de una API, cuyo proceso implica la ejecución de varios pasos. Estos dependen de los requisitos específicos de la tecnología que se aplique estos de forma general son los siguientes:

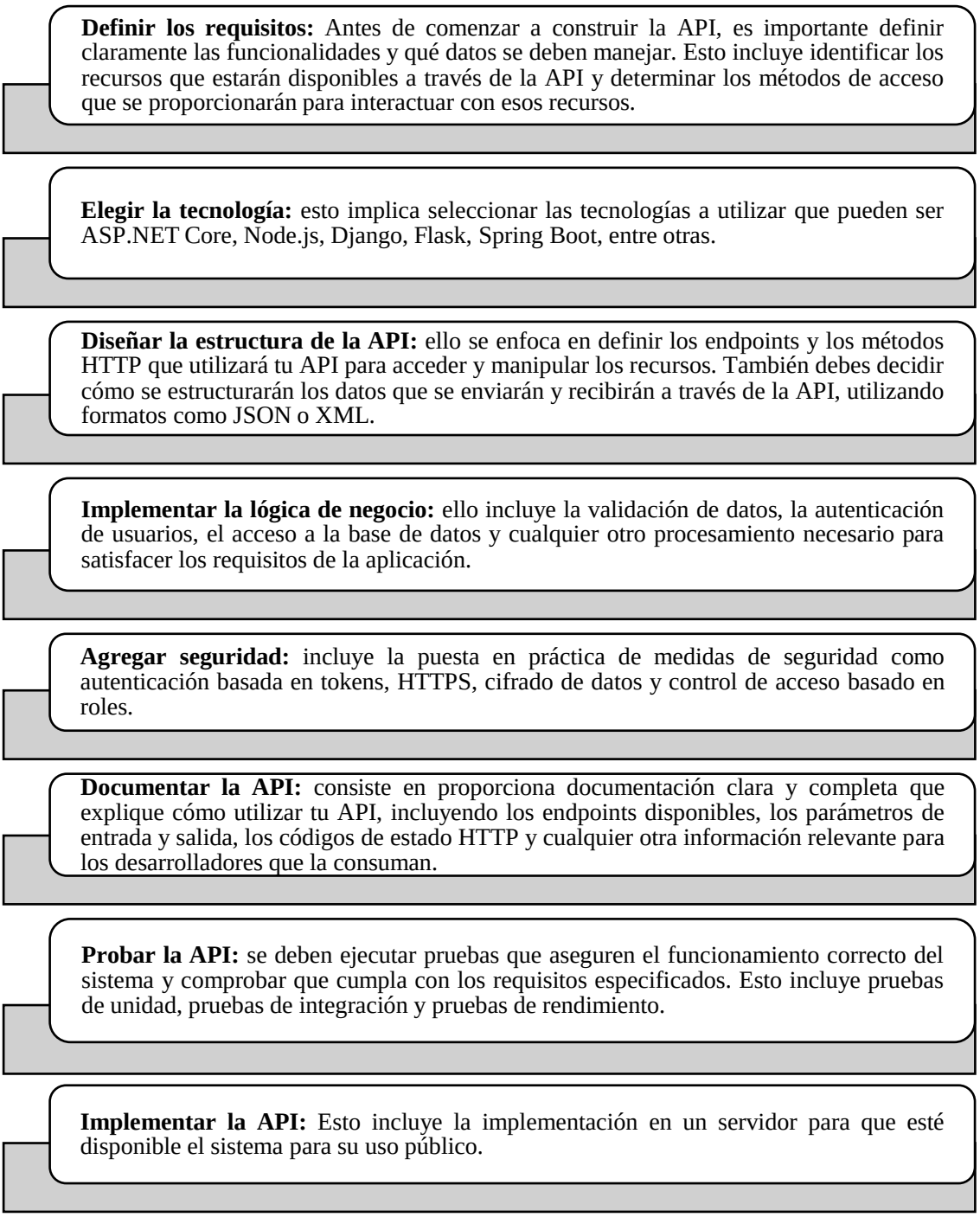


Figura 8.Pasos para crear una API.
Fuente: (Puerta, 2018)

5.6.2. Endpoint

Corresponde referirse al *endpoint* que tal como lo indica su nombre, es un punto final de comunicación en una API (Interfaz de Programación de Aplicaciones). Este especifica una

dirección única a la que se pueden enviar las solicitudes y desde la cual se pueden recibir las respuestas (Hernández J. , 2017).

En el contexto de las *APIs web*, un *endpoint* se refiere a una URL específica que está asociada con una operación o función particular de la API. Cada punto suele estar relacionado con una acción específica que puede realizar el cliente de la API, como recuperar datos, enviar datos, actualizar recursos, eliminar recursos, entre otros (Hernández J. , 2017).

Se debe exponer que, los *endpoints* son fundamentales en el diseño y la implementación de una API, ya que definen cómo interactuar con los recursos y servicios proporcionados por la API. Este tiene una URL única y puede admitir diferentes métodos HTTP, como *GET, POST, PUT y DELETE*, para realizar operaciones específicas en los datos.

5.6.3. Métodos HTTP

Los métodos HTTP son acciones que se pueden realizar en recursos web. Estos métodos definen la acción que se debe realizar en el recurso especificado (Salas, 2022). A continuación, se describen los métodos HTTP más comunes:

- *GET*: Se utiliza para recuperar datos del servidor. Cuando se realiza una solicitud, el servidor devuelve los datos solicitados en el cuerpo de la respuesta. Este método no debe tener efectos secundarios en el servidor y se debe utilizar para operaciones de solo lectura.
- *POST*: Se emplea para enviar datos al servidor para crear un nuevo recurso. Por lo general, se utiliza para enviar datos de formularios al servidor. Los datos enviados con una solicitud POST generalmente se encuentran en el cuerpo de la solicitud y se utilizan para crear un nuevo recurso en el servidor.

- *PUT*: Se hace uso de ello para enviar datos al servidor para actualizar un recurso existente o crear un recurso si no existe. A diferencia de POST, PUT se considera idempotente, lo que significa que realizar la misma solicitud PUT varias veces tendrá el mismo resultado que realizarla una vez.
- *DELETE*: Se utiliza para eliminar un recurso del servidor. Al enviar una solicitud a un recurso específico, el servidor eliminará ese recurso si existe.
- *PATCH*: Se emplea para realizar modificaciones parciales en un recurso. A diferencia de PUT, que reemplaza completamente un recurso, PATCH se utiliza para realizar modificaciones específicas en un recurso existente.
- *HEAD*: Se utiliza para solicitar los encabezados de respuesta que se recibirían si se realizara una solicitud GET a un recurso, pero sin el cuerpo de la respuesta.
- *OPTIONS*: Sirve para solicitar información sobre las opciones de comunicación disponibles para un recurso o servidor. Por ejemplo, los navegadores web a menudo envían solicitudes OPTIONS antes de realizar solicitudes cruzadas de origen (CORS).

5.6.4. Swagger

Corresponde definir que el *Swagger* es una herramienta de código abierto que se utiliza para describir, documentar y probar APIs de manera sencilla y eficiente. Esta permite crear documentación interactiva para sus APIs de una manera fácil de entender (De Santiago, 2023). Entre sus principales características se identifican las siguientes:

- Descripción de la API: Swagger permite describir cada aspecto de su API, incluyendo los endpoints disponibles, los parámetros de entrada y salida, los códigos de estado HTTP y cualquier otro detalle relevante (De Santiago, 2023).
- Documentación interactiva: se puede generar automáticamente documentación interactiva para su API, que incluye descripciones detalladas de cada endpoint, ejemplos de solicitud

y respuesta, y la capacidad de probar la API directamente desde la documentación (De Santiago, 2023).

- Generación de código cliente: puede generar automáticamente código cliente para diferentes lenguajes de programación a partir de la descripción de la API, lo que facilita a los desarrolladores consumir la API desde sus aplicaciones (De Santiago, 2023).
- Validación y prueba: esto incluye herramientas para validar y probar APIs, lo que permite a los desarrolladores asegurarse de que su API funciona correctamente y cumple con los requisitos especificados (De Santiago, 2023).

5.7. Control de versiones

El control de versiones es un sistema que registra los cambios realizados en un conjunto de archivos a lo largo del tiempo, permitiendo mantener un historial de modificaciones, recuperar versiones anteriores, y coordinar el trabajo entre múltiples personas en un proyecto de desarrollo de *software* u otro tipo de proyecto (Sepúlveda, 2023).

En ese orden, el control de versiones es primordial en el desarrollo de *software* por varias razones. En primer lugar, facilita la gestión del historial al mantener un registro detallado de todas las modificaciones realizadas en los archivos a lo largo del tiempo. Esto resulta útil para recuperar versiones anteriores en caso de necesidad, lo que garantiza un seguimiento completo de los cambios realizados.

Además, el control de versiones permite el trabajo colaborativo al permitir que múltiples personas trabajen en los mismos archivos de forma simultánea. Esto se logra coordinando los cambios y fusionando las contribuciones de manera eficiente, lo que facilita la colaboración y la comunicación entre los miembros del equipo (Sepúlveda, 2023).

Otra ventaja importante del aspecto estudiado es la capacidad de experimentación y ramificación que ofrece el control de versiones. Esto se traduce en la posibilidad de crear ramas (*branches*) para probar nuevas funcionalidades o correcciones de errores sin afectar la rama principal (*trunk*) del proyecto, lo que permite un desarrollo más ágil y seguro.

Adicionalmente, el control de versiones proporciona una forma estructurada de revisar y auditar los cambios realizados en el código. Esto se logra mediante la asociación de comentarios, etiquetas y metadatos a cada modificación, lo que facilita la revisión y mejora la calidad del código (Sepúlveda, 2023).

Por último, el control de versiones ayuda a manejar los conflictos que pueden surgir cuando dos o más personas realizan cambios simultáneos en el mismo archivo. Proporciona herramientas para resolver estos conflictos de manera efectiva, lo que evita problemas y asegura la integridad del código.

Asimismo, entre los sistemas de control de versiones más utilizados se encuentra *Git*, que es ampliamente reconocido por su flexibilidad y eficiencia en la gestión de cambios en proyectos de desarrollo de software. Otros ejemplos incluyen *SVN (Subversion*), Mercurial y *CVS (Concurrent Versions System)*, cada uno con características específicas que se adaptan a diferentes necesidades y contextos de desarrollo.

5.7.1. Git

Se debe comenzar por exponer que, *Git* es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux y desde entonces ha ganado una gran popularidad en la

comunidad de desarrollo de software debido a su eficiencia, flexibilidad y potentes características (Paz, 2017).

Por otro lado, existen un grupo de conceptos de *Git* que son de primordial importancia definir para para entender su funcionamiento y utilidad en el desarrollo de software. En primer lugar, es un sistema de control de versiones distribuido, lo que significa que cada usuario tiene una copia completa del repositorio de código en su propia máquina. Esto permite trabajar de manera independiente y sin conexión a internet, en contraste con los sistemas centralizados donde todos los cambios se almacenan en un único servidor (Paz, 2017).

Igualmente, está la definición de *commit* que es esencial en *Git*, ya que representa un conjunto de cambios realizados en los archivos del proyecto. Cada *commit* se acompaña de un mensaje descriptivo que explica los cambios efectuados. Esto facilita la comprensión del historial de modificaciones y la colaboración entre desarrolladores (Paz, 2017) .

Por otro lado, están las ramas (*branches*) que son fundamentales en el *Git* ya que permite la creación de ramas independientes del flujo principal de desarrollo (Paz, 2017). Esto facilita el trabajo en nuevas funcionalidades o correcciones de errores sin afectar el código en producción, y las ramas pueden fusionarse, posteriormente con la principal, para integrar los cambios.

De igual modo, las fusiones (merges) son herramientas esenciales en *Git* para combinar cambios entre ramas, lo que posibilita la integración del trabajo realizado en diferentes ramas en una sola rama. Esto resulta útil para incorporar nuevas funcionalidades desarrolladas en paralelo o para aplicar correcciones de errores a la rama principal de manera ordenada (Paz, 2017).

Finalmente, *Git* facilita la colaboración entre múltiples desarrolladores mediante repositorios remotos en servidores compartidos. Estos permiten enviar y recibir cambios realizados localmente. Todo ello, facilita el trabajo en equipo y la coordinación de proyectos distribuidos de manera efectiva.

5.7.2. Github

Github es una plataforma en línea fundada en 2008 que proporciona servicios de alojamiento para proyectos de desarrollo de software basados en el sistema de control de versiones Git. Esta se ha convertido en un espacio muy popular para que los desarrolladores compartan y colaboren en proyectos de código abierto y privados (Moure, 2023). Entre las características de ella se encuentran la posibilidad de crear repositorios públicos y privados, donde los usuarios pueden alojar sus proyectos y controlar quién puede ver o contribuir al código (Hinojosa & Merelo, 2017).

Por su parte, *Github*, ofrece herramientas completas para el control de versiones con *Git*, lo que permite a los desarrolladores realizar *commits*, crear ramas, fusionar cambios y gestionar el historial de versiones de sus proyectos de manera eficiente (Hinojosa & Merelo, 2017).

La colaboración es otro aspecto destacado de *Github*, ya que facilita la interacción entre desarrolladores al permitirles clonar repositorios, enviar solicitudes de extracción (*pull requests*), revisar el código de otros desarrolladores, realizar comentarios y discutir cambios propuestos. Esto fomenta un entorno de trabajo colaborativo y ágil (Moure, 2023).

Además, *Github* ofrece herramientas para la gestión de problemas y el seguimiento de errores. Los usuarios pueden reportar problemas, realizar un seguimiento del progreso de las

tareas, asignar tareas a otros miembros del equipo y mantener conversaciones sobre problemas específicos, lo que facilita la organización y el mantenimiento de los proyectos (Moure, 2023).

Por último, *Github* se integra de manera fluida con diversas herramientas de desarrollo de software populares, como IDEs, sistemas de integración continua, herramientas de revisión de código y sistemas de gestión de proyectos. Esta integración facilita la sincronización y coordinación del flujo de trabajo de desarrollo, mejorando la eficiencia y la calidad del proceso de desarrollo de software.

5.8. IDE (Entorno de desarrollo integrado)

El Entorno de desarrollo integrado (IDE) es un software que proporciona un conjunto de herramientas y funciones integradas para facilitar el desarrollo de software. Este incluye un editor de código, herramientas de compilación y depuración, así como funciones para administrar proyectos y controlar versiones (Moreno, 2018).

Se debe decir, que algunos IDEs tiene características adicionales como resaltado de sintaxis, completado automático de código, sugerencias de código, integración con sistemas de control de versiones, generación de documentación, pruebas unitarias y soporte para diferentes lenguajes de programación (Moreno, 2018).

De igual forma, Los IDEs son herramientas muy útiles para los desarrolladores, ya que les permiten escribir, probar y depurar código de manera más eficiente en un entorno integrado y centralizado. Esto ayuda a mejorar la productividad y la calidad del software desarrollado. Ejemplos de IDEs populares incluyen *Visual Studio, IntelliJ IDEA, Eclipse y PyCharm*.

5.8.1. Visual Studio Code

Corresponde plantar que, *Visual Studio Code* es un editor de código fuente desarrollado por Microsoft. Aunque su nombre incluye "*Visual Studio*", se debe aclarar, que no es lo mismo que el IDE completo *Visual Studio*. En cambio, *Visual Studio Code* es más ligero y está diseñado principalmente como un editor de texto avanzado con características específicas para el desarrollo de software (Gamarra, 2023).

Por otro lado, *Visual Studio Code* es altamente personalizable y admite una amplia variedad de lenguajes de programación gracias a su arquitectura de complementos. Ofrece características como resaltado de sintaxis, completado automático, refactorización de código, integración con sistemas de control de versiones (como *Git*), depuración integrada, terminal integrada y una amplia gama de extensiones que pueden instalarse para adaptar el entorno según las necesidades del desarrollador (Gamarra, 2023).

De forma general se puede afirmar que *Visual Studio Code* es una herramienta muy popular entre los desarrolladores debido a su rapidez, facilidad de uso y soporte para múltiples plataformas. Esta se encuentra disponible para *Windows, macOS* y *Linux*. Además, al ser de código abierto, permite a la comunidad contribuir con extensiones y mejoras al editor.

5.8.2. Visual Studio 2022

Se debe revisar el *Visual Studio 2022*, que es una versión del popular entorno integrado de desarrollo (IDE) desarrollado por Microsoft. Es la última versión de la serie *Visual Studio*, diseñada para ayudar a los desarrolladores a escribir código, depurar y compilar aplicaciones para una variedad de plataformas y tecnologías (Ockert, 2022).

Por otro lado, *Visual Studio 2022* ofrece una serie de características y mejoras significativas que benefician a los desarrolladores en su trabajo diario. En primer lugar, se han realizado mejoras significativas en el rendimiento y la productividad del entorno de desarrollo (Ockert, 2022). Esto se traduce en un mayor rendimiento al escribir código y trabajar en proyectos de cualquier tamaño, lo que proporciona una experiencia más fluida y eficiente.

Además, *Visual Studio 2022* brinda un amplio soporte para tecnologías modernas y diversas plataformas, abarcando desde aplicaciones de escritorio hasta aplicaciones web, móviles, desarrollo de juegos y más. Esta versatilidad permite a los desarrolladores trabajar en una variedad de proyectos sin limitaciones tecnológicas.

Asimismo, la integración con Azure es otra característica destacada de Visual Studio 2022. Proporciona herramientas integradas que facilitan el desarrollo, la implementación y el mantenimiento de aplicaciones en la nube de *Microsoft Azure*. Esto simplifica el proceso de trabajar en entornos de computación en la nube (Ockert, 2022). En cuanto a la depuración y el análisis de código, Visual Studio 2022 según este autor, incluye herramientas avanzadas que ayudan a encontrar y corregir errores de manera más eficiente. Esto mejora la calidad del código y acelera el proceso de desarrollo.

Finalmente, *Visual Studio 2022* ofrece opciones de personalización y extensibilidad. Los desarrolladores pueden personalizar el entorno de desarrollo mediante la instalación de extensiones y la configuración de ajustes según sus necesidades específicas, lo que les permite adaptar *Visual Studio* a sus flujos de trabajo y preferencias individuales.

5.9. Patrones de arquitectura de software

En relación con los patrones de arquitectura de software se debe decir que constituyen soluciones típicas a problemas comunes que se encuentran en el desarrollo de software. Estos representan las mejores prácticas que se han utilizado por desarrolladores experimentados. Estos operan como plantillas que se adaptan a las necesidades particulares de un problema específico durante la programación. Estos son de gran valor porque facilitan la reutilización de soluciones probadas y efectivas, y ayudan a estructurar su código de manera más eficiente (Bascón, 2017).

La importancia de los patrones de arquitectura está en su capacidad de proporcionar un lenguaje común entre los desarrolladores. Esto facilita la comunicación dentro de los equipos de desarrollo, permitiendo que los diseñadores y programadores discutan soluciones de manera más efectiva y eficiente. Ellos mejoran la reusabilidad del código. Al emplear estructuras de diseño estandarizadas, los desarrolladores pueden reutilizar soluciones que han sido efectivas en proyectos anteriores. Esto no solo reduce el tiempo de desarrollo, sino que también mejora la coherencia y la confiabilidad del código al basarse en métodos que han sido depurados y probados en múltiples contextos (Pavón, 2023).

Los patrones de diseño también fomentan la escalabilidad y la mantenibilidad del software. Al seguir estos patrones, el código tiende a ser más organizado, modular y adaptable, facilitando la gestión de grandes bases de código y la incorporación de nuevas funcionalidades (Pavón, 2023). Por ello, son fundamentales para el desarrollo moderno de aplicaciones, entre ellos están el patrón MVVM, MVP y el MVC, este último aplicado en este trabajo.

5.9.1. MVVM (Model-View-ViewModel)

El patrón de diseño Model View ViewModel (MVVM) es una estructura arquitectónica que mejora la separación de la lógica de la interfaz gráfica del usuario (GUI) de la lógica de negocio y los datos, siendo especialmente popular en aplicaciones con interfaces ricas y dinámicas. Su estructura se compone por el modelo que representa los datos y la lógica de negocio de la aplicación. Es responsable de gestionar las reglas de negocio, las operaciones de datos, y las interacciones con las bases de datos, asegurando que los datos sean correctos y estén actualizados (Gavilánez y Layedra, 2022).

Por su lado, la vista se ocupa de la presentación y la interacción con el usuario. Esta muestra la información procedente del ViewModel en un formato adecuado para la interacción, capturando cualquier entrada de usuario y mostrando los resultados del procesamiento de estos datos. Por último, el ViewModel, sirve de puente entre la vista y el modelo, gestionando la lógica de la vista. Transforma los datos del modelo para que puedan ser fácilmente presentados por la vista y responde a comandos y acciones del usuario, actualizando el modelo según corresponda (Gavilánez & Layedra, 2022).

Cabe agregar que entre sus principales ventajas están la separación de responsabilidades, la facilidad de pruebas, que el data binding se utiliza en el enlace de datos (data binding), lo que elimina la necesidad de manipular la interfaz de usuario de manera imperativa, entre otros.

5.9.2. MVP (Model-View-Presenter)

El patrón Modelo-Vista-Presentador (MVP) es una arquitectura de diseño empleada en el desarrollo de software para separar claramente la lógica de la interfaz de usuario de la lógica de negocio. Este patrón es útil en aplicaciones que requieren una distinción nítida entre cómo se

presentan los datos y cómo se gestionan, común en aplicaciones web y móviles (González, 2023).

En ese sentido, MVP contiene los datos y la lógica de negocio, manejando operaciones como las consultas a bases de datos y el mantenimiento de las reglas de negocio. La vista se encarga de la presentación y la interacción con el usuario, pero a diferencia de otros patrones como MVVM, la vista en MVP no tiene conocimiento directo del modelo y solo interactúa con el presentador.

Por su lado, el presentador funciona como un intermediario entre la vista y el modelo; procesa la entrada de la vista, manipula los datos del modelo según sea necesario y luego pasa estos datos de vuelta a la vista en un formato adecuado para su presentación. Se debe exponer que entre sus ventajas están la separación de responsabilidades, que facilita el mantenimiento y la gestión del código. También el desacoplamiento de la interfaz de usuario del modelo.

5.9.3. MVC (Model View Controller)

En ese marco, específicamente el patrón de diseño MVC, utilizado en este estudio se divide el sistema en tres componentes principales: el modelo, la vista y el controlador. Cada uno de estos se encapsula en objetos distintos. Ello para garantizar que las responsabilidades estén claramente separadas y que no se mezclen en una sola clase. Esto permite organizar mejor la información, la lógica del sistema y la interfaz de usuario (Hernández R. , 2021).

Por su lado, el modelo es el componente encargado de manejar los datos y las operaciones relacionadas con la base de datos. Su función es puramente gestionar dichos datos, y no incluye ninguna lógica relacionada con la presentación de estos datos al usuario. Asimismo, la vista,

muestra la información al usuario, lo que permite dar a conocer los datos existentes en el sistema. Lo anterior, no interpreta el significado de la información ni las posibles interacciones del usuario con ella.

Por su parte, el controlador actúa como el intermediario que gestiona las entradas del usuario. Es responsable de recibir estas entradas, procesarlas adecuadamente y, en coordinación con el modelo y la vista, asegura que los datos necesarios se manipulen correctamente para luego ser presentados por la vista.

Por lo anterior, los componentes antes descritos, están intrínsecamente conectados. En resumen, la vista despliega la información del modelo al usuario, el controlador interpreta las acciones del usuario y actualiza el modelo según sea necesario, y cualquier cambio en el modelo se refleja de nuevo en la vista, actualizando la información mostrada al usuario.

Se debe mencionar, por ser el patrón utilizado en esa investigación, que tiene varias ventajas entre las que se destacan que las vistas de la aplicación siempre presentan información actualizada. Además, que los desarrolladores no necesitan gestionar manualmente la actualización de las vistas, ya que el modelo de la aplicación se encarga automáticamente de este proceso. De igual forma que cualquier cambio necesario en el modelo del dominio, como la adición de métodos o datos, solo requiere modificaciones en el modelo y su interfaz con las vistas, sin necesidad de alterar todo el sistema de comunicación y actualización entre modelos (Bascón, 2017).

Igualmente, las alteraciones realizadas en las vistas no impactan otros módulos de la aplicación. Por sus ventajas es ampliamente adoptado en la actualidad dentro de frameworks de

programación orientada a objetos diseñados para el desarrollo de aplicaciones de gran escala; tecnologías como Java Swing, Apache Struts, Microsoft ASP.NET. Este es eficaz y proporciona, a las aplicaciones que lo implementan una capacidad de extensión y mantenimiento superior en comparación con aquellas basadas en otros patrones.

5.10. Clean Code

El concepto de clean code (Código Limpio) se refiere a una práctica de programación que se centra en la escritura de código que no solo funcione, sino que sea fácil de leer, entender y modificar. La importancia de mantener su aplicación va más allá de la estética; ya que es fundamental para la eficiencia y la sostenibilidad a largo plazo de los sistemas (Espinosa, 2018).

En esa línea, un código limpio se caracteriza por su claridad, simplicidad y la ausencia de duplicidades. Al evitar la redundancia en el código, se reduce la probabilidad de errores y se facilita la gestión de las dependencias, lo que a su vez hace que el código sea menos propenso a problemas en el futuro. Además, al estar bien estructurado y limpio es más fácil de testear y depurar, ya que las anomalías son más detectables y localizables cuando el código es directo y sin complicaciones innecesarias (Molina, 2021).

Otra ventaja del clean code, es que mejora la colaboración entre desarrolladores, ya que, al ser más fácil de entender, permite el trabajo en equipo ya que todos los desarrolladores pueden comprender y modificar el código con mayor facilidad. Asimismo, ahorra tiempo y mejora la calidad del software porque lo hace más eficiente y menos susceptible a errores costosos.

6. Desarrollo del Proyecto de Titulación

La metodología de desarrollo que se utilizó en esta investigación fue MVC. Esta ha dado forma al diseño, la implementación y las pruebas del sistema de condominio SCC para aplicar en el condominio “Plaza 10”. Dicha metodología de desarrollo es ordenada y eficiente proporcionando un marco de trabajo estructurado que permitió desarrollar, probar e implementar el proyecto en cuestión.

Igualmente, el uso de la metodología MVC ha permitido promover un desarrollo estructurado y eficiente del sistema, facilitando una separación clara de responsabilidades entre la lógica de negocio, la interfaz de usuario y el control de flujo de datos. Esto ha garantizado la implementación de mejoras constantes en las funcionalidades del sistema durante su desarrollo. Esta elección metodológica ha permitido optimizar el proyecto y establecer una base sólida para su mantenimiento, impulsando una entrega puntual y exitosa.

6.1. Aspectos teóricos vinculados al diseño e implementación del proyecto

El primer objetivo trazado en esta investigación es el siguiente: “Identificar y enunciar los aspectos teóricos que se relacionan con las herramientas necesarias para el diseño e implementación del proyecto prototipo para la gestión de condominios”.

Este objetivo ha sido desarrollado y cumplido en este trabajo, a partir de que se ha realizado un estudio teórico, tanto del condominio desde su definición, clases y otros como del desarrollo web a partir de los elementos que se han aplicado en el sistema. A respecto se realizó una revisión bibliográfica y documental, donde se recopiló y analizó la información existente relacionada con el tema de estudio. Ello incluyó la consulta de libros, artículos científicos, y otras fuentes relevantes.

Luego se sintetizó y analizó la información teórica recopilada. Todo ello permitió elaborar el marco teórico donde se integraron y expusieron de manera sistemática los principales conceptos que se abordan en el problema de investigación. Todo ello ha establecido en este estudio, una base sólida de conocimiento, sustentada en la literatura existente sobre el tema de estudio.

6.2. Resultados encuestas gestión de condominios

En relación con el segundo objetivo de esta investigación se debe plantear que fue trazado de la siguiente forma: “Analizar las necesidades del edificio Plaza 10 en cuanto a la gestión de condominios, identificando las funcionalidades claves que debe ofrecer el sistema”.

Este se llevó a la práctica a partir de la aplicación de una encuesta como instrumento de investigación, la que se adjunta en el anexo 1 de este trabajo y que está conformada por un grupo de preguntas. Se debe destacar que este instrumento es confiable, flexible y permite obtener los resultados del estudio de manera rápida.

Asimismo, mediante la aplicación de la encuesta, se recopilaron datos directamente de la población objetiva. Ello proporcionó, a su vez, la información precisa sobre las necesidades del condominio objeto de estudio. Seguidamente, se ilustran los resultados obtenidos al respecto.

En ese orden, para aplicar las encuestas, se inició por definir la población objetiva conformada por 274 personas en su condición de residentes, guardias de seguridad y directivos. Luego a través del cálculo de la muestra mediante la aplicación de una formula estadística, que se expone a continuación, se determinó la muestra que fue objeto de este instrumento.

$$n = \frac{m}{e^2(m-1)+1}$$

n = tamaño de la muestra (?)

m = población y universo (274)

e = porcentaje de error (5%)

Por lo tanto, la muestra con la que se trabajó corresponde a 163 personas.

Los resultados arrojados por dicho instrumento de investigación fueron los siguientes a partir de las preguntas del cuestionario elaborado y aprobado que se detalla en el anexo 1 de este estudio:

Pregunta 1: ¿Es usted propietario o arrendatario de alguna unidad del edificio Plaza 10?

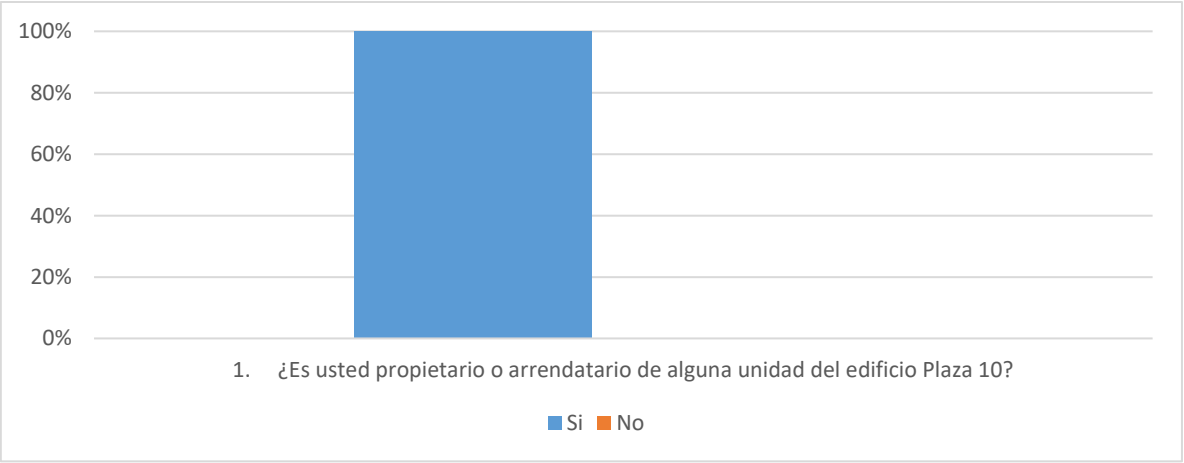


Figura 9.Muestra resultados pregunta 1 del cuestionario.
Elaborado por el autor.

El gráfico muestra que todos los encuestados son propietarios o arrendatarios de alguna unidad en el edificio Plaza 10. Ello demuestra una alta implicación de los residentes en la comunidad.

Pregunta 2: ¿Cuáles considera usted que son los problemas más comunes dentro del condominio?

Los resultados arrojados fueron los siguientes:

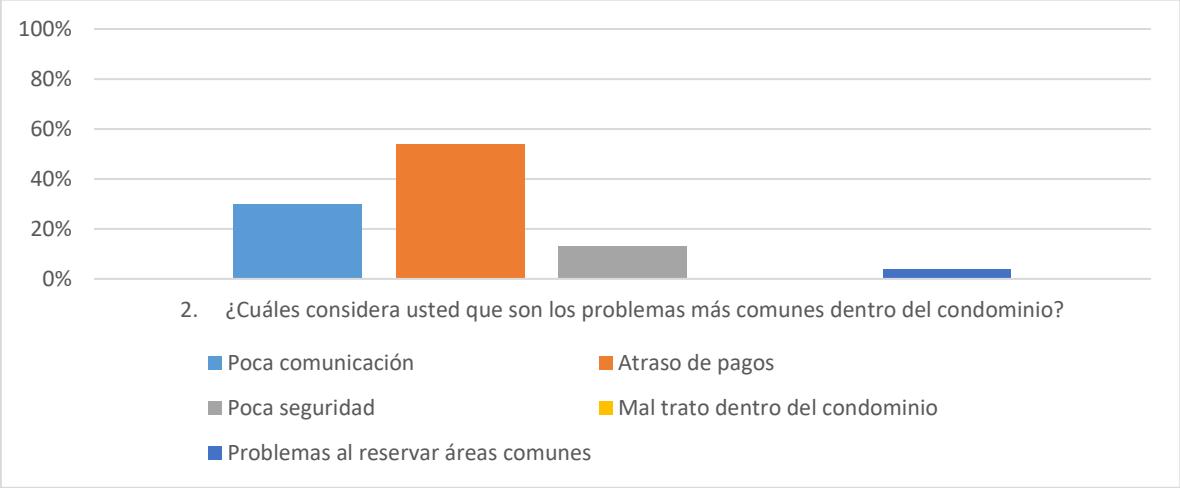


Figura 10.Muestra resultados pregunta 2 del cuestionario.
Elaborado por el autor.

En relación con lo ilustrado en la figura, queda claro que, la mayoría de los encuestados (53%) identifican el atraso de pagos como el problema más común, seguido de la poca comunicación (30%) y la falta de seguridad (13%). Esto indica áreas clave que pueden necesitar atención y que se pueden solucionar con la aplicación de un sistema de condominios en el edificio Plaza 10.

Pregunta 3: ¿Por cuál medio se informa usted de las reuniones, mingas y eventos del condominio?

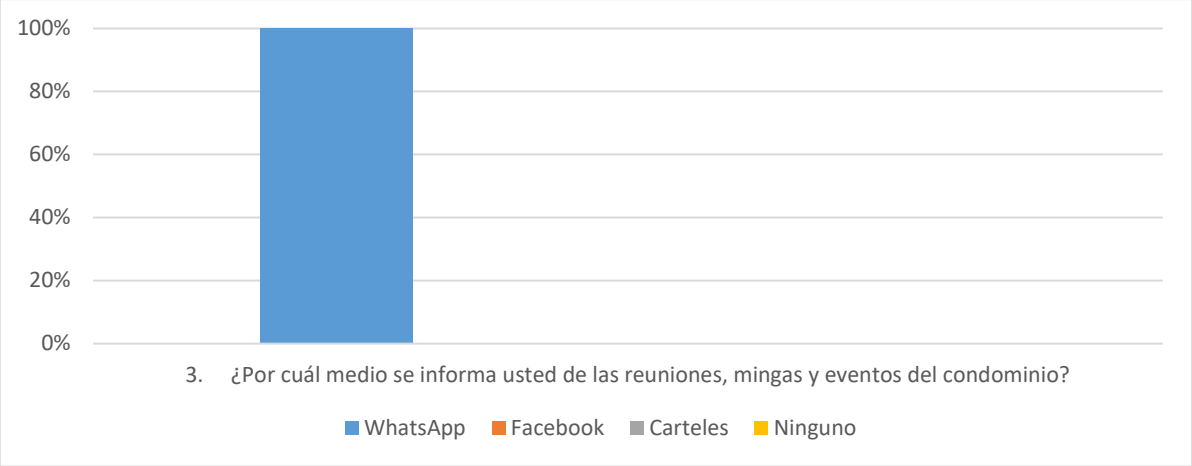


Figura 11. Muestra resultados pregunta 3 del cuestionario.
Elaborado por el autor.

Tal como muestra la figura anterior, todos los encuestados se informan a través de WhatsApp. Esto sugiere que esta plataforma es la más efectiva para comunicar eventos y reuniones dentro del condominio.

Pregunta 4: ¿Usted ha experimentado dificultades al reservar áreas comunes como la sala de reuniones, parrilla, parqueadero de visitas, etc.?

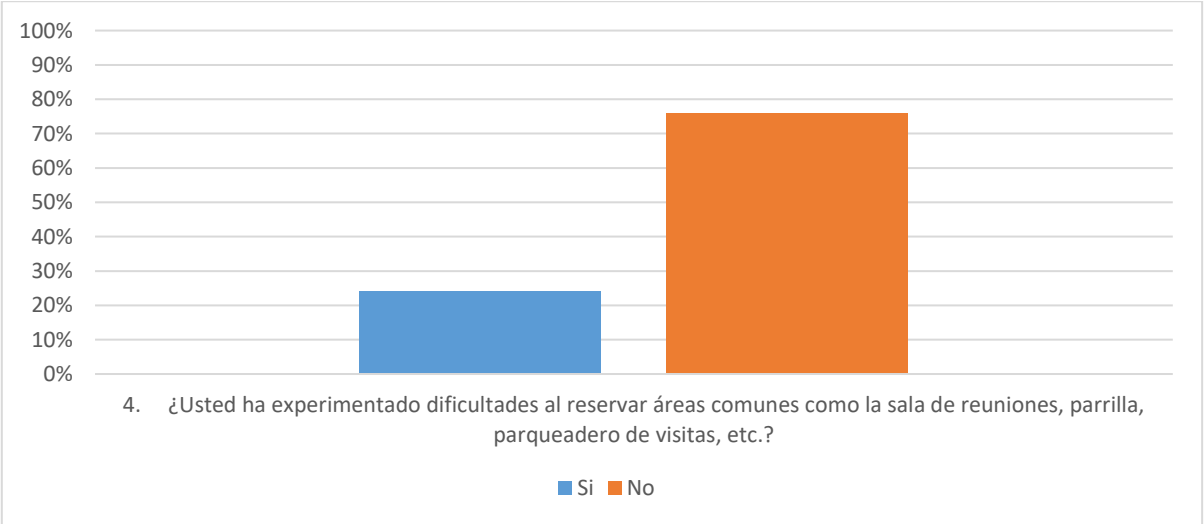


Figura 12. Muestra resultados pregunta 4 del cuestionario.
Elaborado por el autor.

Como resultados de la pregunta 4 del cuestionario, como se muestra en la figura, el 24% de los encuestados ha experimentado dificultades al reservar áreas comunes, lo que señala una

oportunidad para mejorar la eficiencia y accesibilidad en el proceso de reserva a través de un sistema de control mucho más organizado y preciso.

Pregunta 5: ¿Prefiere usted que la red social cuente con un sistema de notificaciones para mantenerlo informado sobre actualizaciones relevantes sobre el condominio?

En relación con esta pregunta los resultados se plasman en la siguiente figura.

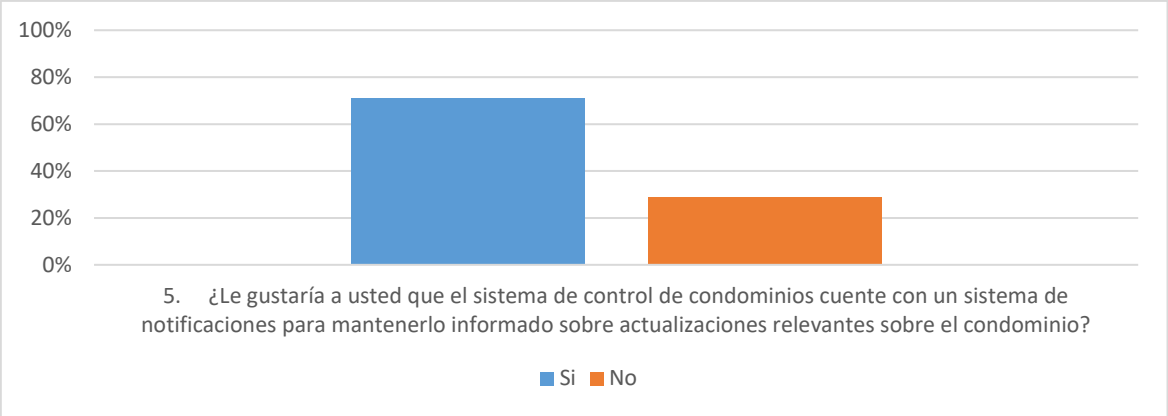


Figura 13. Muestra resultados pregunta 5 del cuestionario
Elaborado por el autor.

Como se aprecia en la figura anterior, la gran mayoría, específicamente el 71% de los encuestados, desean un sistema de notificaciones para mantenerse informados sobre actualizaciones relevantes del condominio. Ello destaca la importancia de la comunicación oportuna y eficaz.

Pregunta 6: ¿Qué dispositivos utilizaría con mayor frecuencia para acceder al sistema de control de condominios?

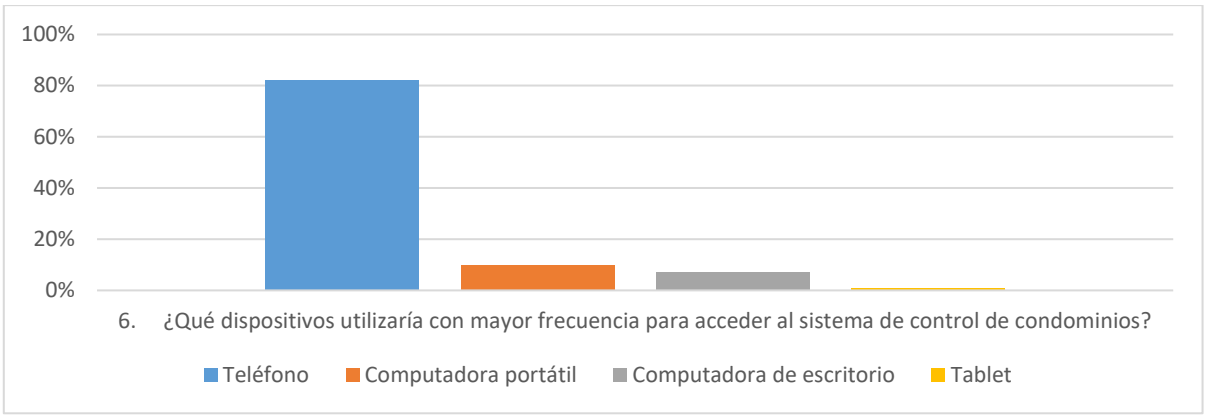


Figura 14.Muestra resultados pregunta 6 del cuestionario.
Elaborado por el autor.

En relación con la pregunta 6, la mayoría de los encuestados, específicamente el 82% contesta que utilizarían su teléfono para acceder al sistema de control de condominios. Esto resalta la importancia de crear un sistema web responsivo que permita la accesibilidad y comodidad al ser utilizado desde un dispositivo celular.

Pregunta 7: ¿Considera usted que una función de mensajería instantánea dentro del sistema de control de condominios facilitaría la comunicación entre miembros del condominio?

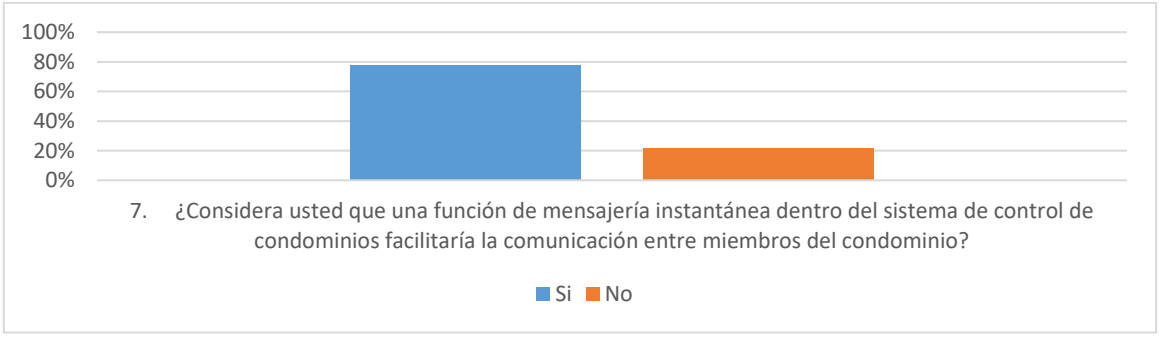


Figura 15.Muestra resultados pregunta 7 del cuestionario.
Elaborado por el autor.

Como ilustra la figura 16, la mayoría (78%) de los encuestados considera que una función de mensajería instantánea facilitaría la comunicación entre los miembros del condominio. Lo anterior sugiere una demanda clara de herramientas de comunicación interna.

Pregunta 8: ¿Cómo preferiría que sea el entorno visual del sistema de control de condominios?

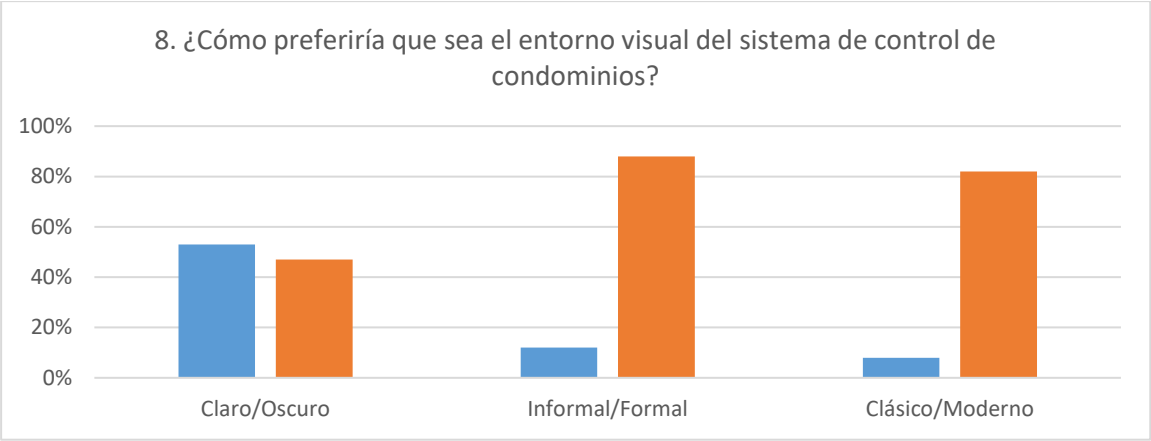


Figura 16.Muestra resultados pregunta 8 del cuestionario.
Elaborado por el autor.

Tal como muestra la figura anterior, la preferencia visual es variada entre los encuestados. Esos muestran una ligera preferencia por un entorno visual claro, formal y moderno.

Pregunta 9: ¿Cuánto estaría usted dispuesto(a) a pagar por tener acceso al sistema de control de condominios?

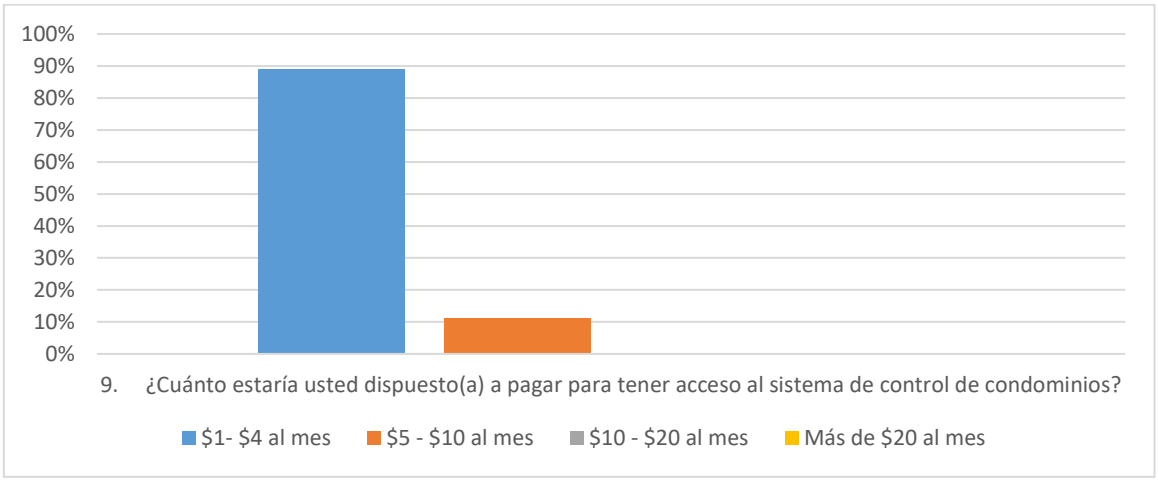


Figura 17.Muestra resultados pregunta 9 del cuestionario.
Elaborado por el autor.

Como se muestra anteriormente, la mayoría, en este caso el 89% de los encuestados estarían dispuestos a pagar entre $1 y $4 al mes por acceso al sistema de control de condominios. Ello indica una disposición general a invertir en esta herramienta.

Pregunta 10: ¿Estaría usted dispuesto(a) a recomendar dicho sistema a un conocido y obtener un descuento del 50% los 6 primeros meses?

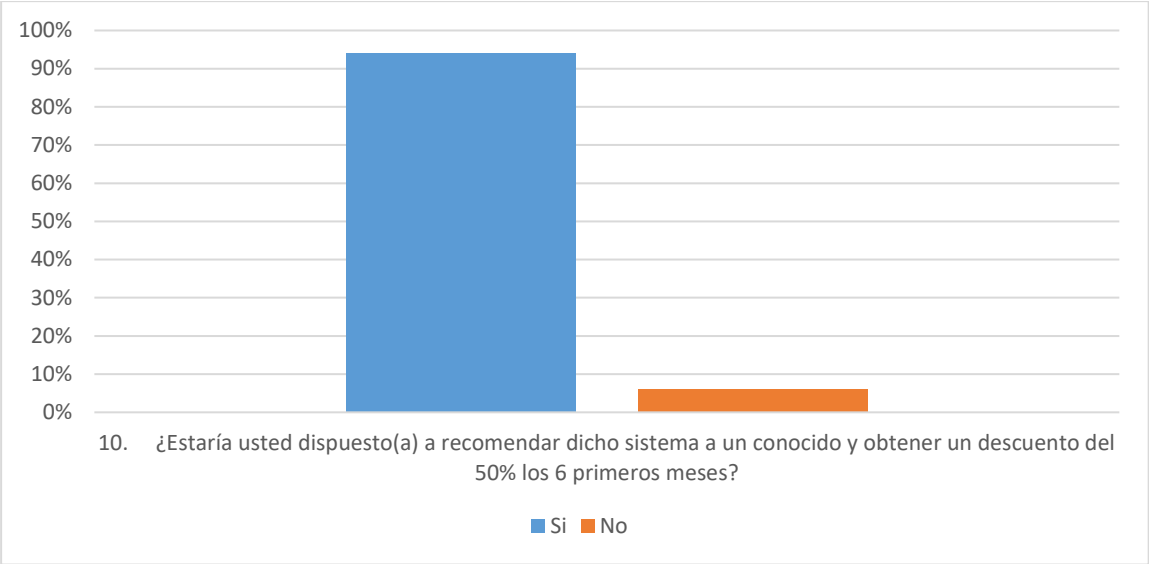


Figura 18.Muestra resultados pregunta 10 del cuestionario.
Elaborado por el autor.

Como ilustra la figura, los encuestados en su gran mayoría (94%) estarían dispuestos a recomendar el sistema a un conocido y obtener un descuento, lo que apunta una satisfacción y confianza en el sistema propuesto.

Pregunta 11: ¿Qué tan interesado estaría en utilizar un sistema de control de condominios que facilitaría y mejoraría la comunicación y organización dentro del condominio?

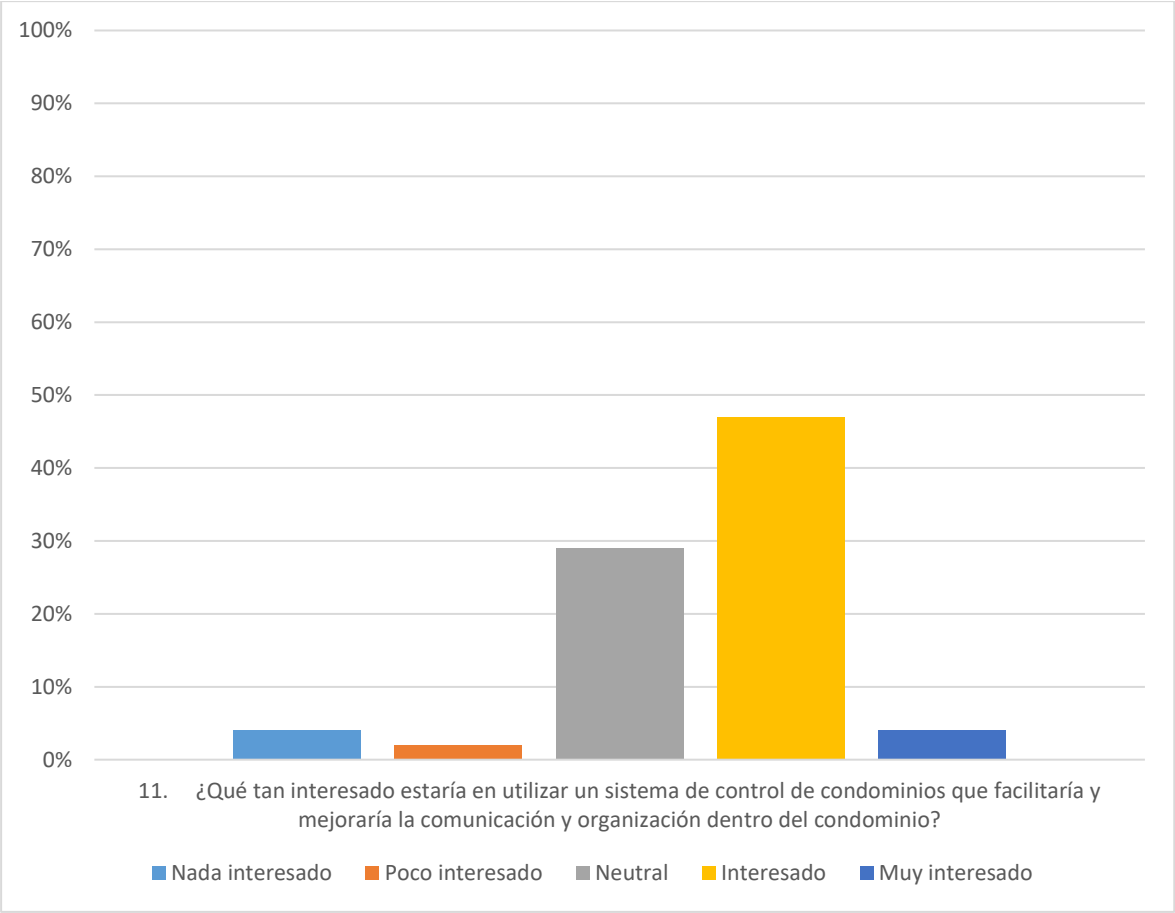


Figura 19. Muestra resultados pregunta 11 del cuestionario.
Elaborado por el autor.

La mayoría de los encuestados están interesados o muy interesados en utilizar un sistema de control de condominios que la convivencia dentro del condominio, lo que refleja una necesidad percibida y un interés en soluciones tecnológicas para mejorar la comunidad.

6.3. Desarrollo sistema de control de condominio

El tercer objetivo formulado para el desarrollo de esta investigación es el siguiente: Desarrollar un sistema de control de condominio para el edificio “Plaza 10” con base a la metodología de desarrollo MVC.

Para dar a conocer este objetivo cumplido se debe partir del siguiente diagrama estructural que muestra las tecnologías y estructura de la aplicación web consistente en un sistema de condominio denominado SCC a aplicar en el edificio antes mencionado:

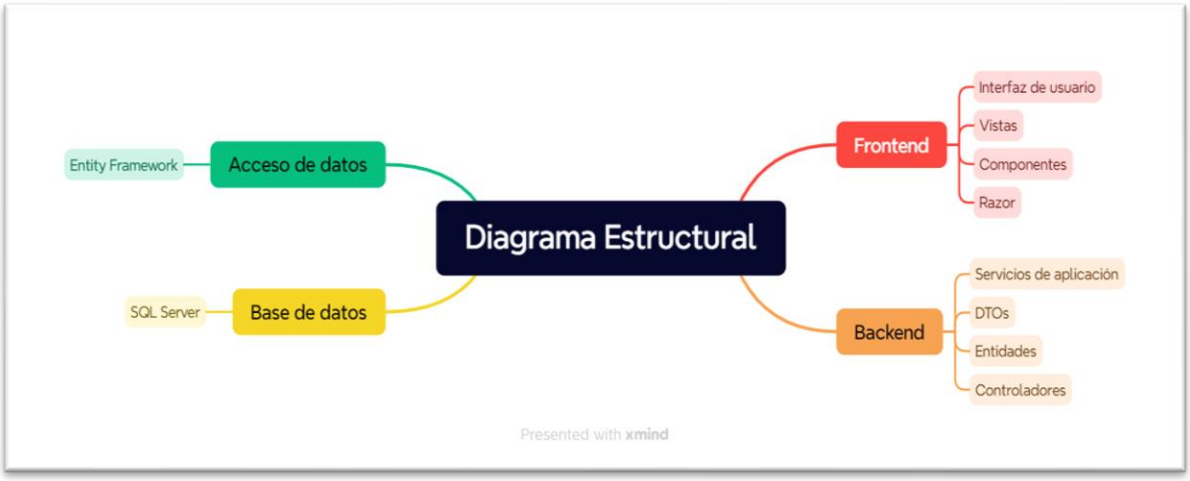


Figura 20. Diagrama Estructural del sistema SCC.
Elaborado por el autor.

Por otra parte, el siguiente diagrama es de comportamiento en que se muestran las funcionalidades que posee el sistema presentado:

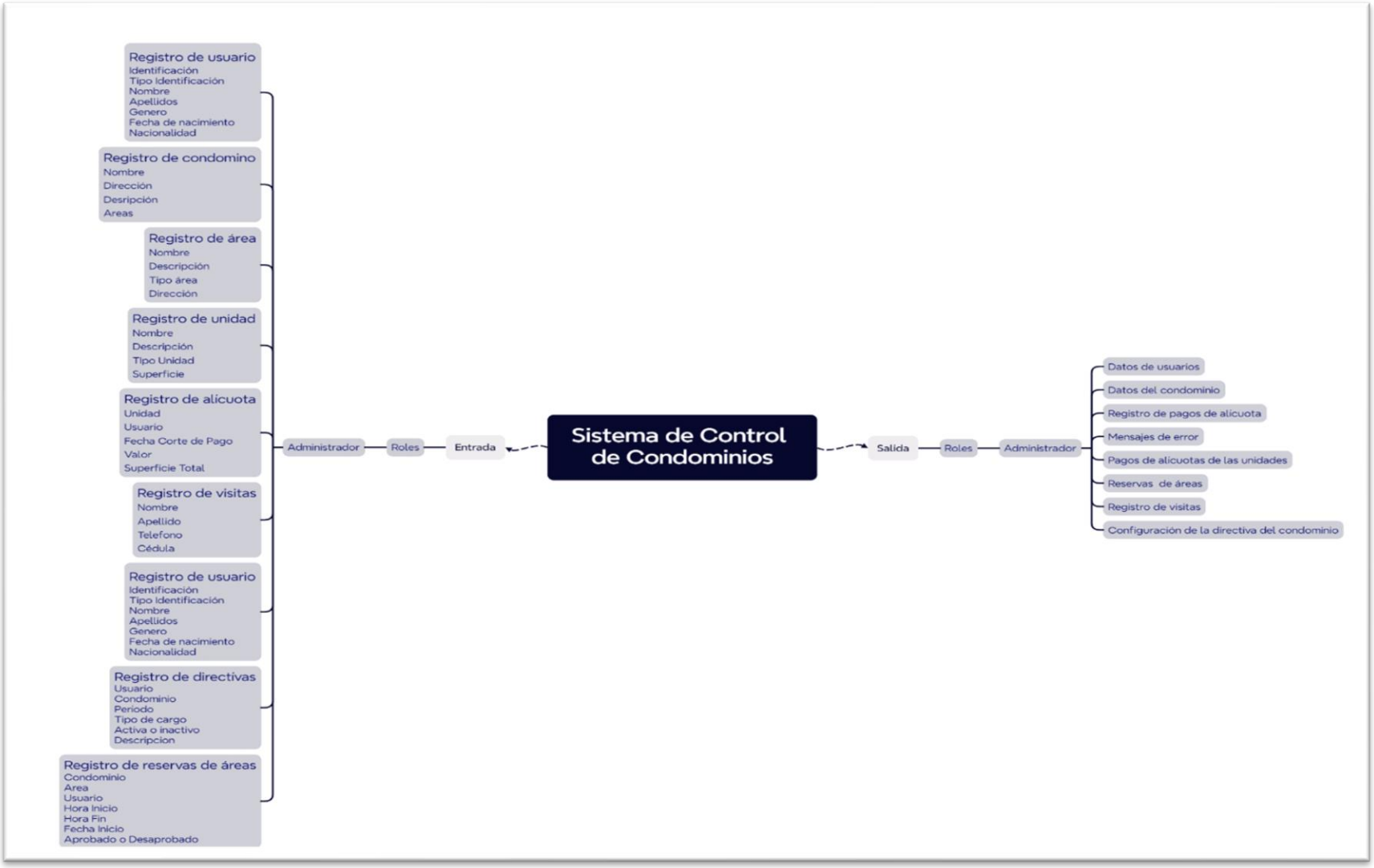


Figura 21.Diagrama de comportamiento.
Elaborado por el autor.

Seguidamente se exponen las funcionalidades de la aplicación web SCC aplicable al edificio Plaza 10.

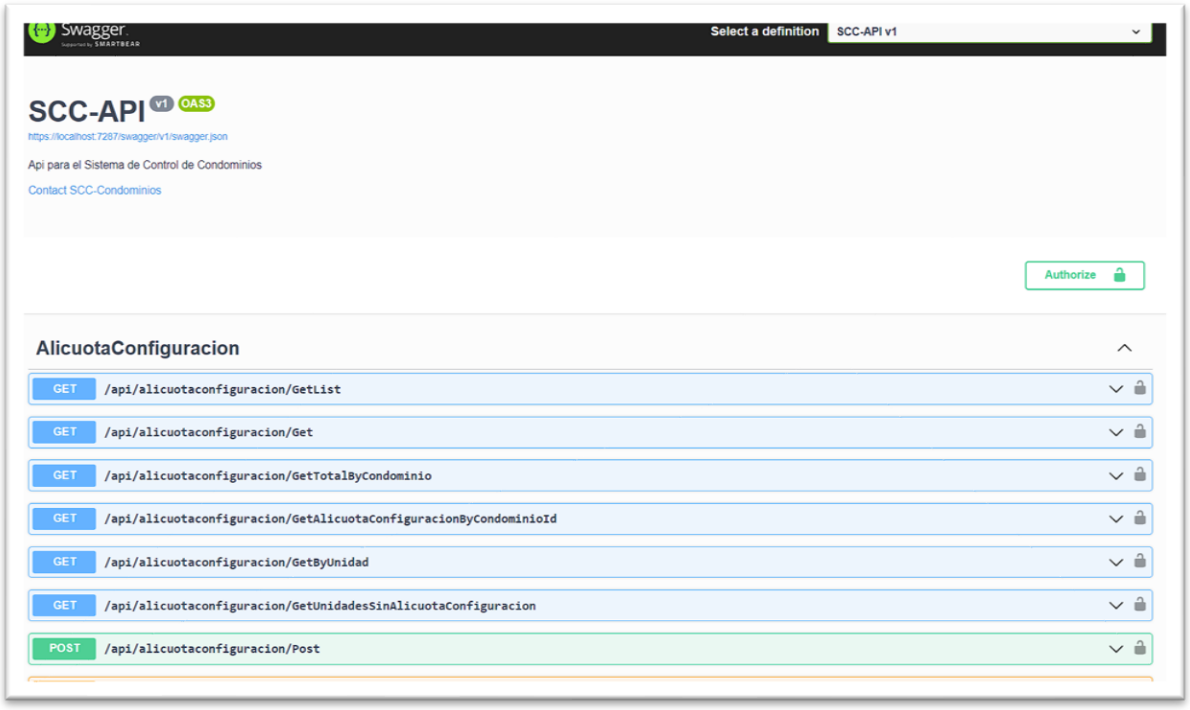


Figura 22.Endpoints de la API.
Elaborado por el autor.

En la anterior figura se muestran algunos de los *Endpoints* utilizados en el sistema para hacer los llamados a la base de datos.

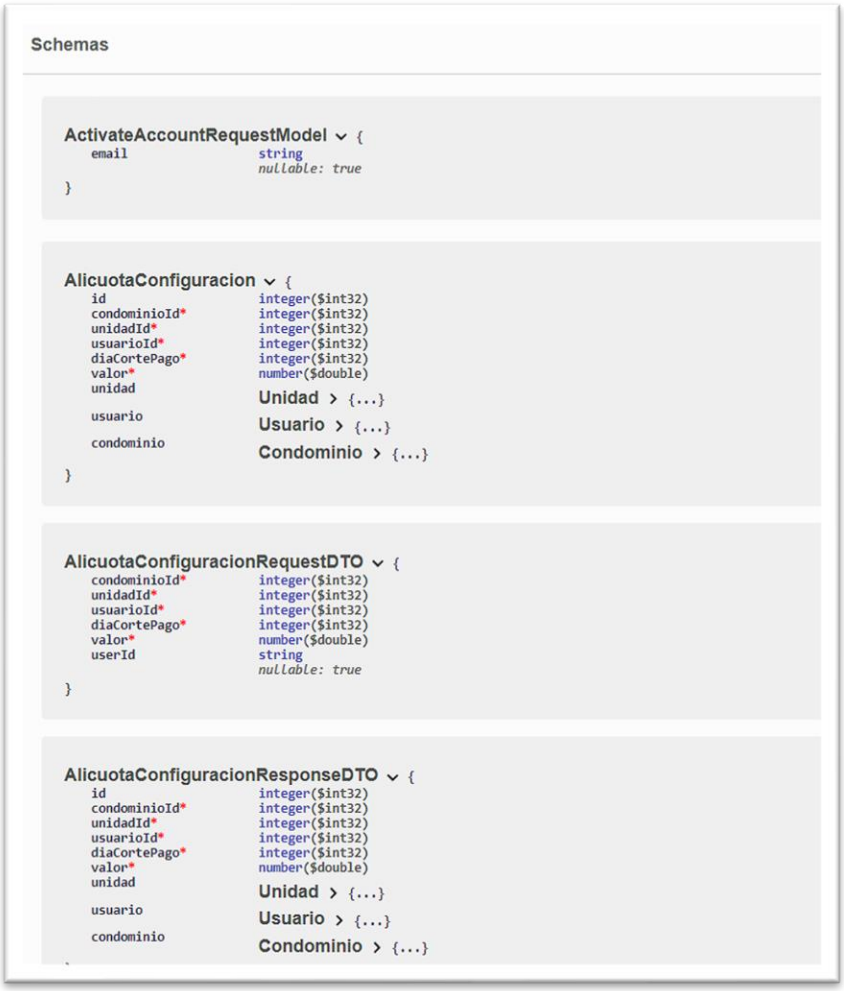


Figura 23.Esquemas de las entidades de la base de datos desde la API.
Elaborado por el autor.

La figura antes expuesta, ilustra los esquemas de las entidades de la base de datos con sus respectivos campos y tipos de variables.

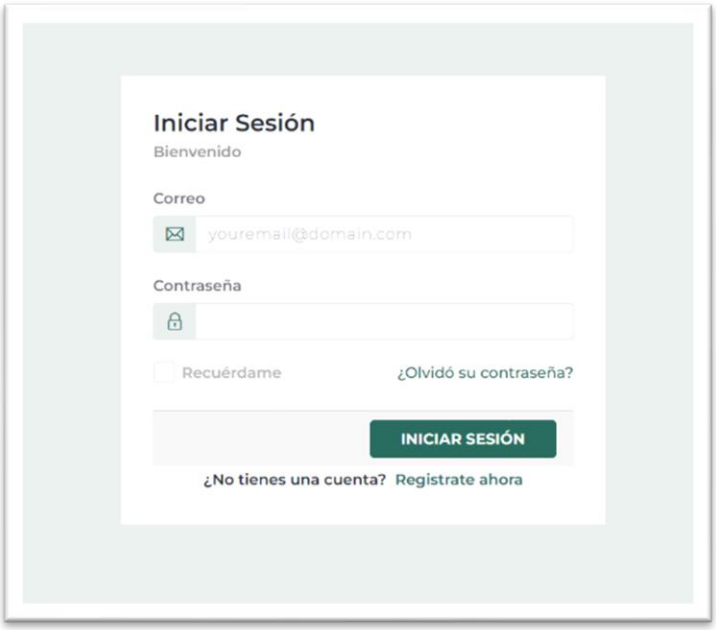


Figura 24. Inicio de Sesión.
Elaborado por el autor.

Como ilustra la figura, este es el formulario para el inicio de sesión al sistema donde se ingresa el correo electrónico del usuario registrado y su respectiva contraseña.

La siguiente figura muestra los campos a llenar para registrarse en el sistema. Para dicho registro se necesitó un correo electrónico, una contraseña y confirmar la contraseña ingresada anteriormente para verificar que se ingresó la contraseña deseada. Luego de haber ingresado los datos necesarios se podrá acceder al sistema de control de condominios.

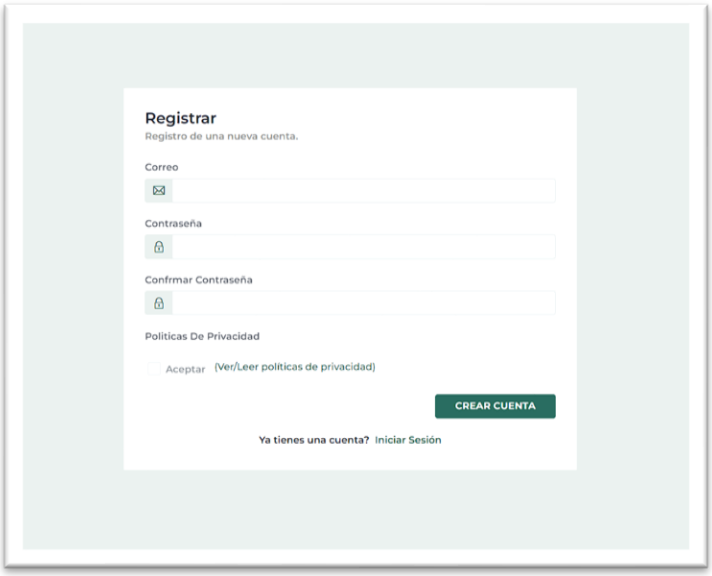


Figura 25.Registro de usuario.
Elaborado por el autor.

A continuación, se muestra en la figura siguiente la vista del *Dashboard*, donde se puede observar los condominios ingresados en el sistema, aparte de diferentes configuraciones que brinda la aplicación web.

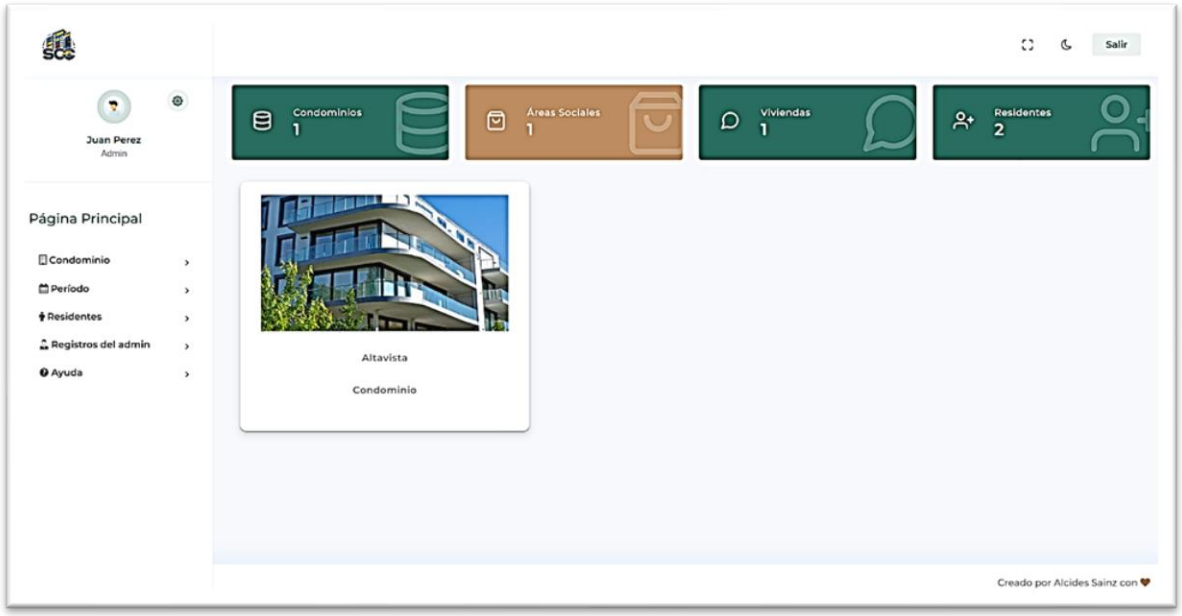


Figura 26. Dashboard de la aplicación web.

Elaborado por el autor.

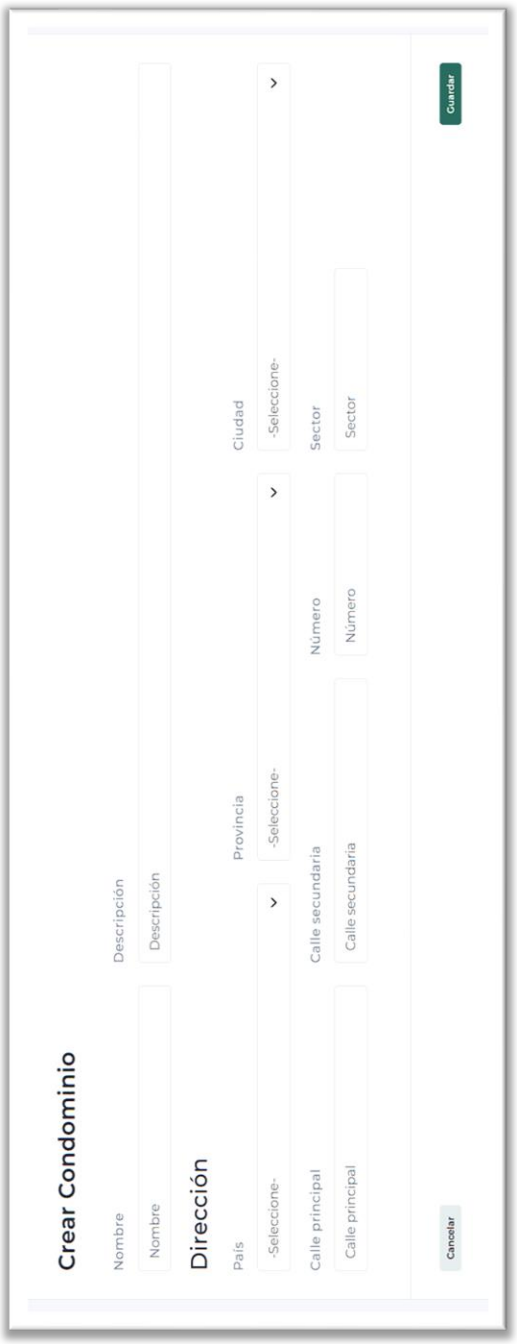


Figura 27.Formulario para crear condominio.
Elaborado por el autor.

La figura anterior muestra un formulario para la creación de un condominio donde se deben ingresar datos como el nombre y la dirección donde se encuentra ubicado.

La figura que se muestra a continuación ilustra la información y los detalles del condominio. Entre ellos expone el nombre y su descripción, su dirección, las diferentes funcionalidades que tienes y las áreas que pertenecen a al condominio seleccionado. Además, cada información se puede editar y eliminar a elección del administrador.

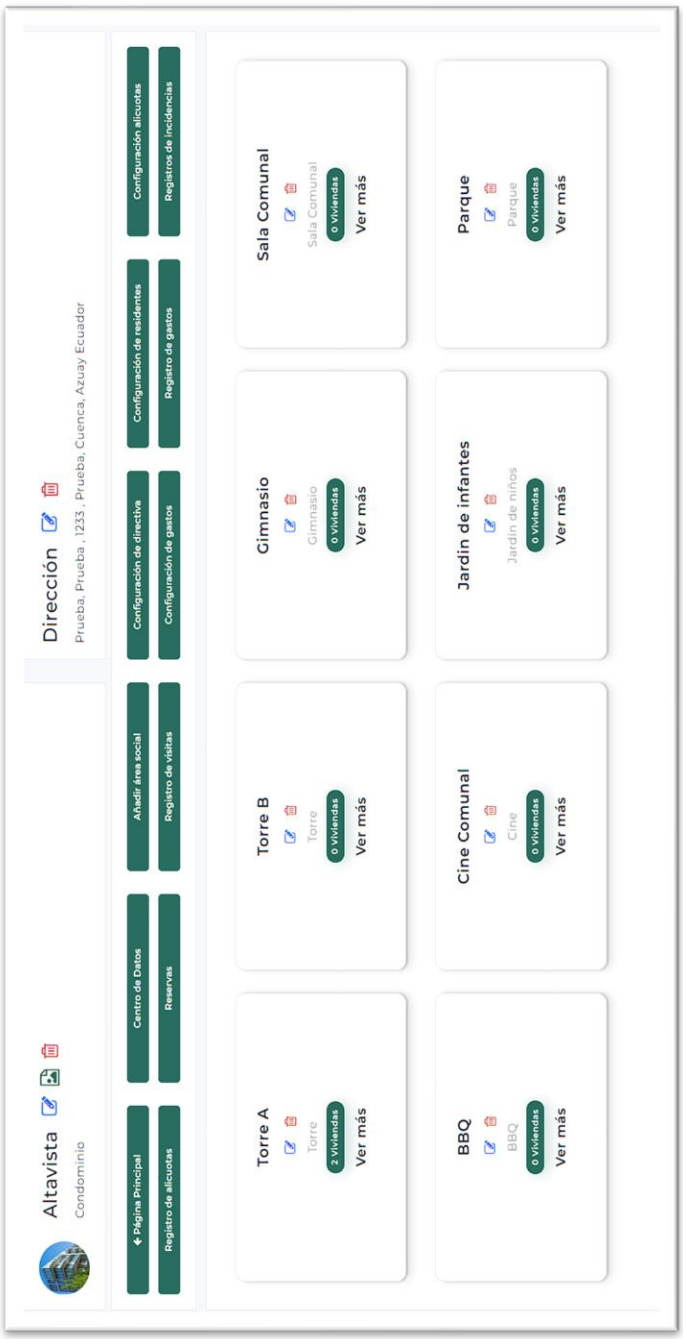


Figura 28.Vista de los datos y funciones del condominio.
Elaborado por el autor.

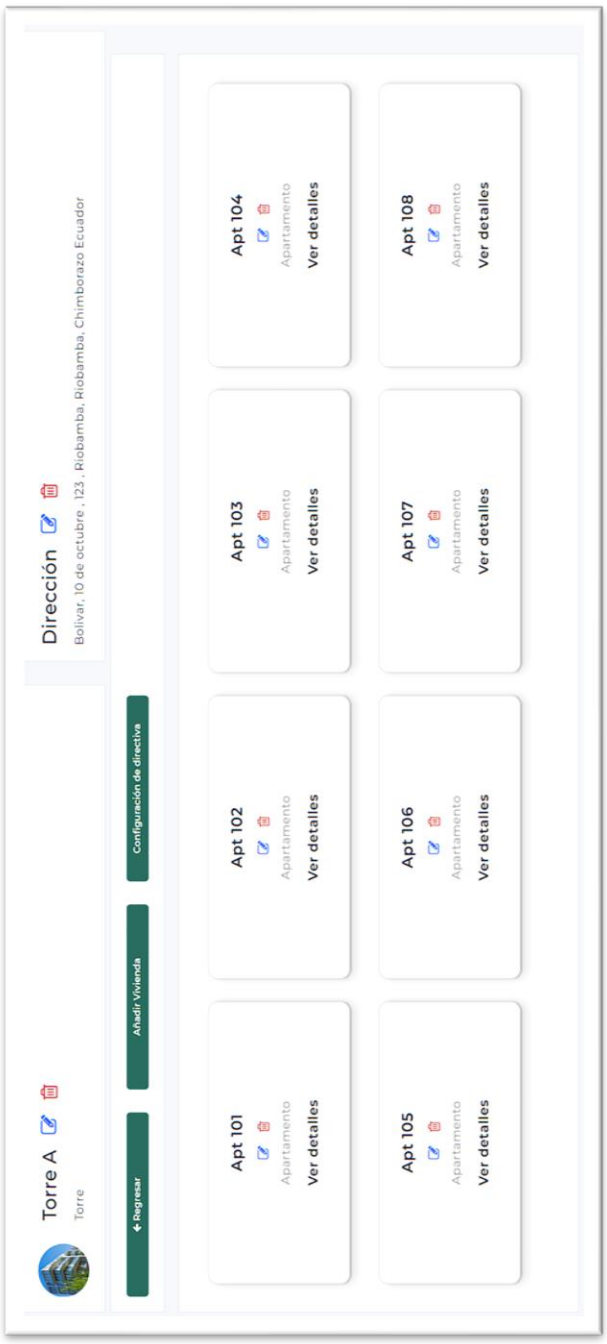


Figura 29. Vista de los datos y funciones del área del condominio.
Elaborado por el autor.

La figura anterior presenta una vista detallada del área. En ella, se visualizan las unidades que pertenecen a dicha área, proporcionando una panorámica completa de la distribución y disposición de las propiedades dentro del condominio. Cada unidad se encuentra claramente identificada, mostrando detalles como su número o nombre, así como el tipo de unidad que es,

ya que podría ser un apartamento, una casa o cualquier otro tipo de vivienda. Además de algunos botones que permiten agregar más unidades o configurar la directiva del área.

En la figura siguiente se observan los usuarios ingresados en el sistema además de poder borras o editar sus datos, También se encuentra un botón para añadir un usuario al sistema.

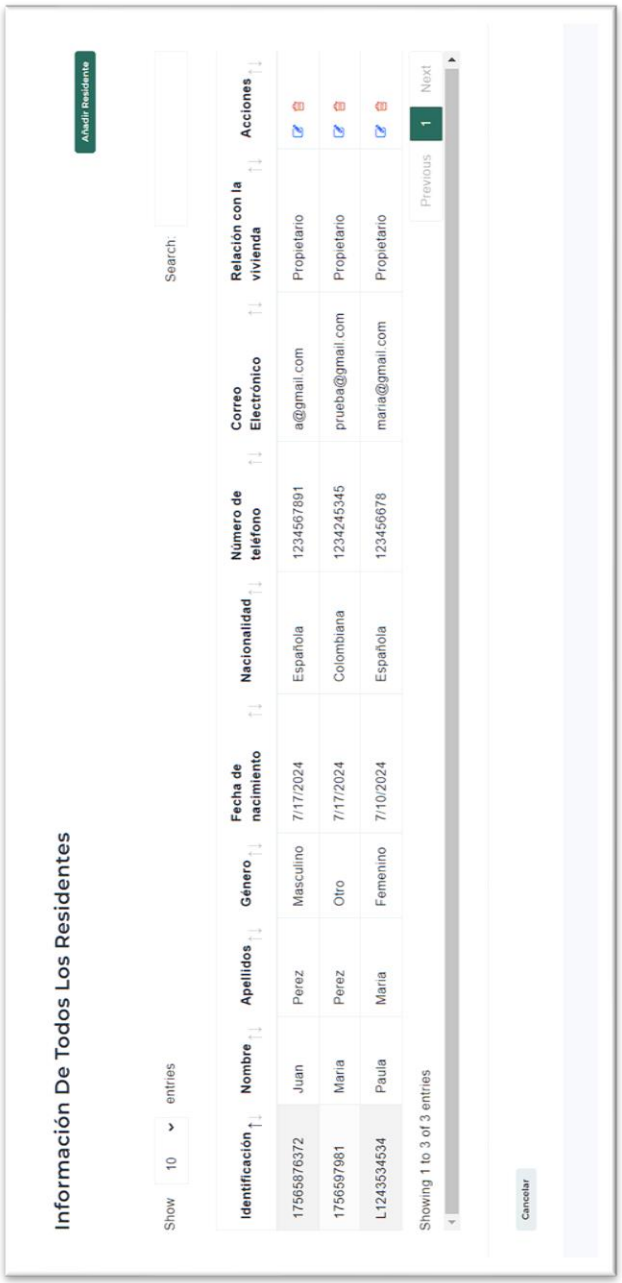


Figura 30. Información de los usuarios ingresados en el sistema.
Elaborado por el autor.

En la figura siguiente se muestran los períodos en que funciona la directiva del condominio. Por ello en el sistema se ilustra la que se asigne por determinado tiempo.

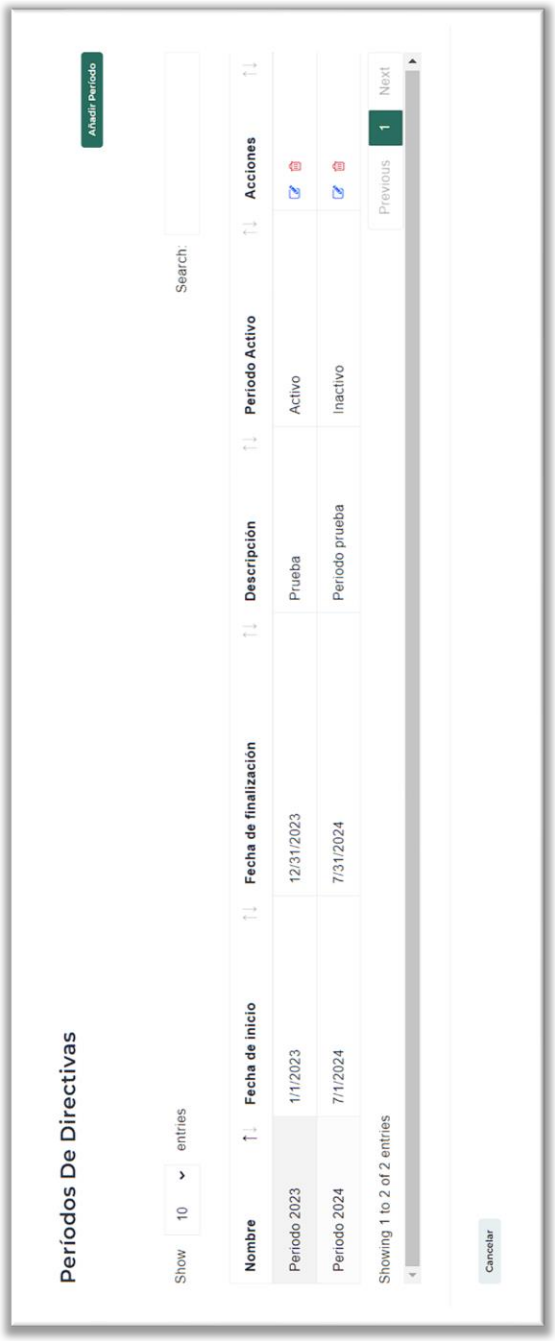


Figura 31. Lista de períodos de directivas del condominio.
Elaborado por el autor.

Asimismo, en la figura que se expone a continuación, se muestra la forma en que se agregan los miembros de la directiva. Ello se realiza utilizando los usuarios ya ingresados en el sistema, para luego seleccionar el tipo de cargo que ocupa esa persona en la directiva.

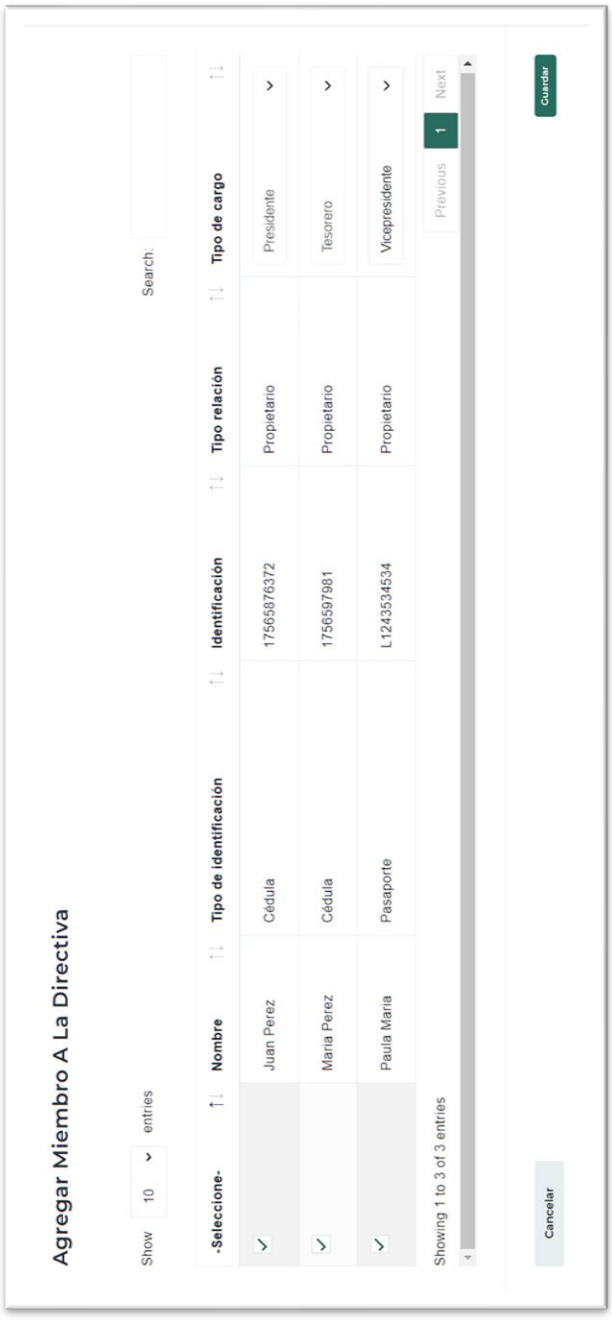


Figura 32. Selección de miembros para la directiva del condominio.
Elaborado por el autor.

En esta figura que sigue se observa, de una forma más ordenada, los miembros de la directiva con su nombre y apellido, su identificación, tipo de cargo. Igualmente, refleja, si actualmente se encuentra activa, dicha directiva o no.

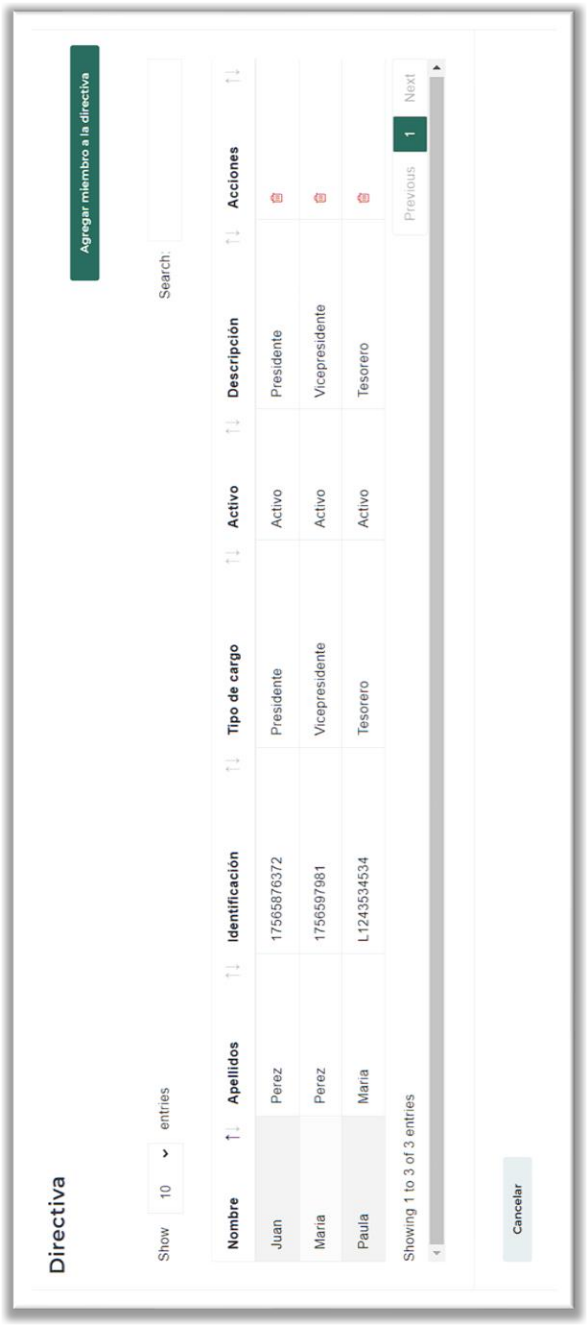


Figura 33. Lista de miembros de la directiva.
Elaborado por el autor.

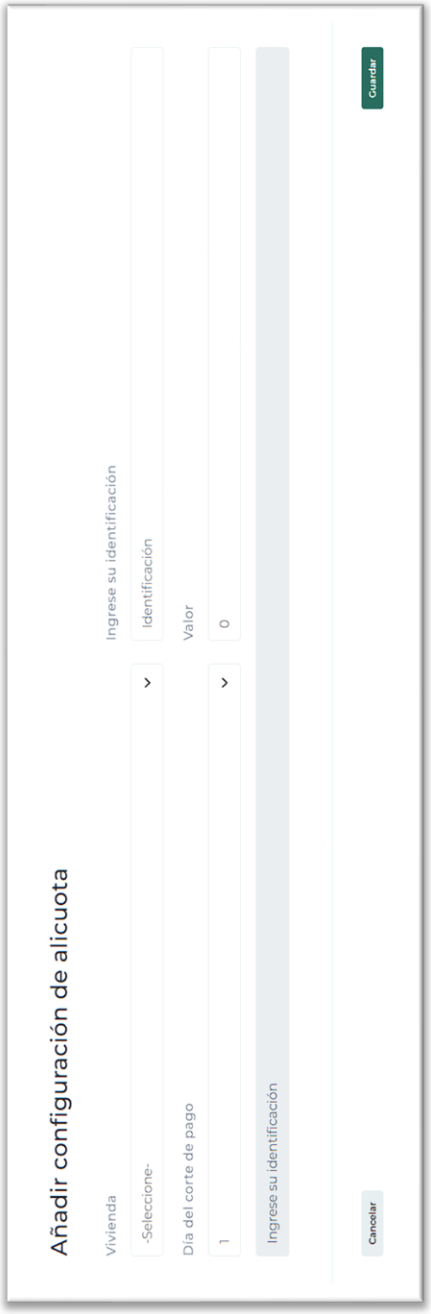


Figura 34. Configuración de alícuota de una unidad.
Elaborado por el autor.

En la anterior figura se configura la alícuota de cada unidad, ingresando la cédula del encargado de esa unidad, el día que en cada mes van a cancelar la alícuota y el valor que se debe de cancelar.

Por su lado, en la vista siguiente, se presentan las unidades ya configuradas. Además, muestra el día de pago de alícuota al igual que el valor que deben de cancelar cada mes.

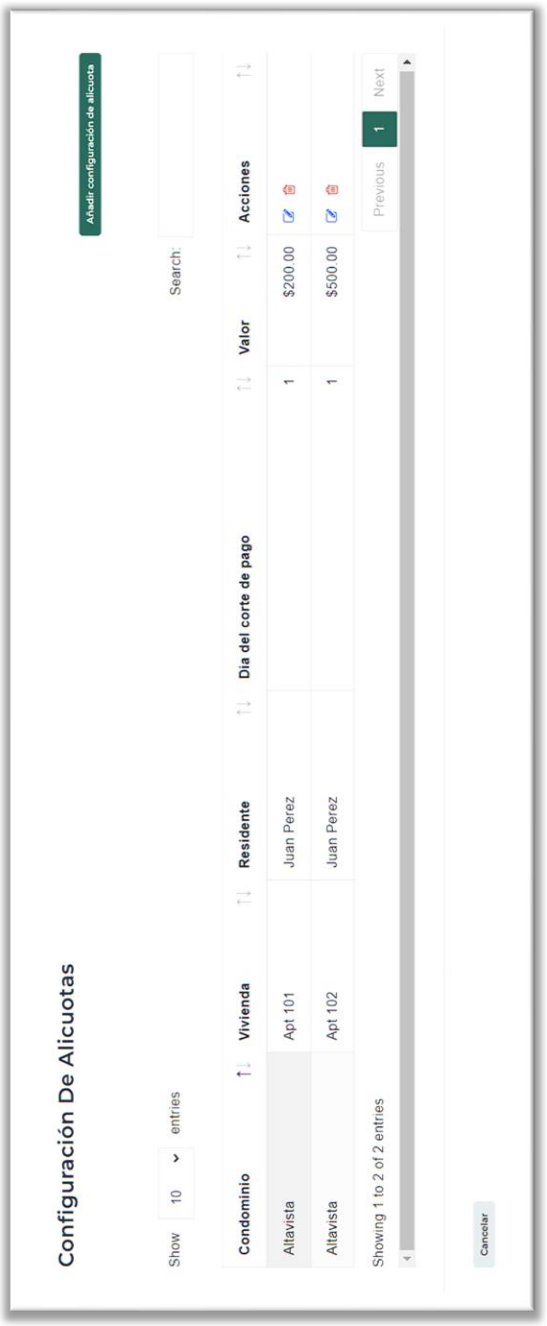


Figura 35.Lista de alícuotas configuradas.
Elaborado por el autor.

Por otra parte, en la figura que sigue, se presenta el registro de todas las alícuotas ya configuradas. A partir de ello, se puede seleccionar el mes y año que se quiere consultar. Esto permite saber que unidad ha realizado el pago de la alícuota y cual está pendiente.

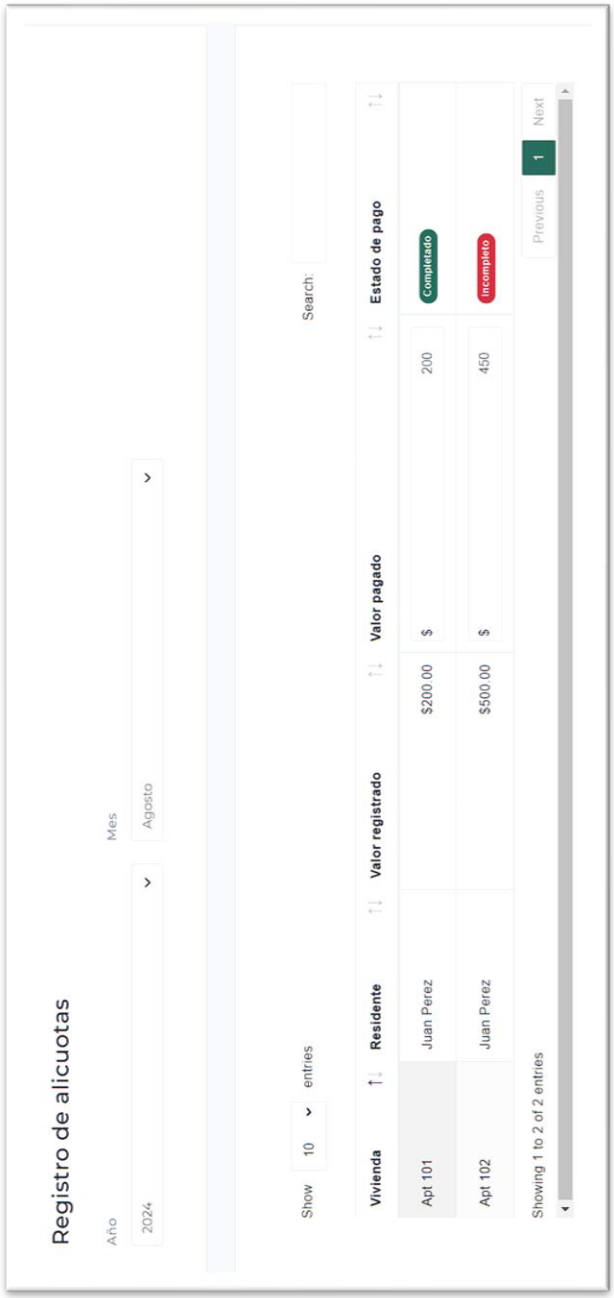


Figura 36.Registro de alícuotas por mes y año
Elaborado por el autor.

En la figura siguiente se expone el formulario para la reserva de áreas.

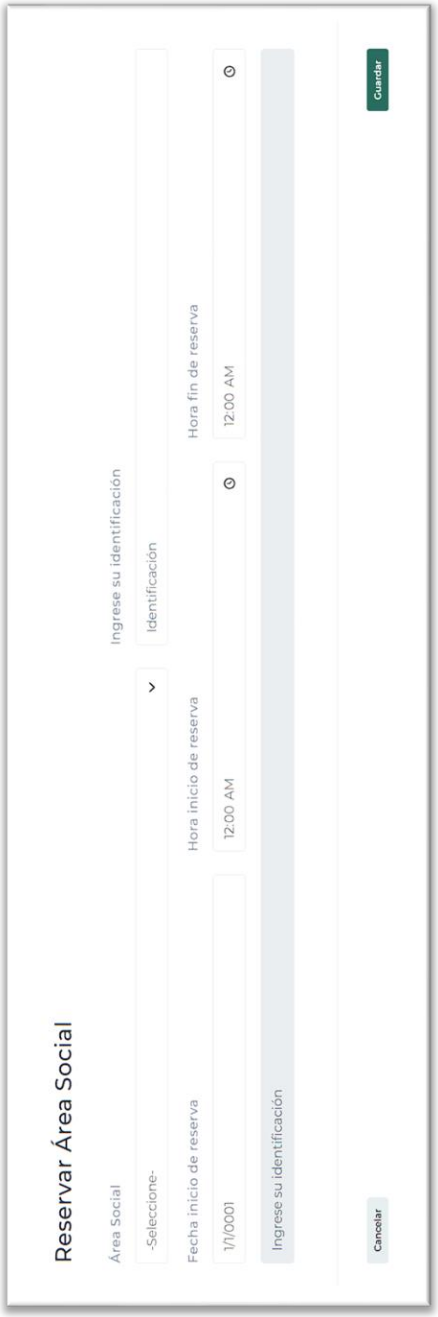


Figura 37. Formulario para la reserva de áreas
Elaborado por el autor.

Asimismo, la figura que sigue expone el registro de aquellas reservas solicitadas para espacios como la sala comunal u otras. En este *DataTable* se pueden visualizar con la finalidad de que el administrador del sistema pueda aprobar cada reserva solicitada.

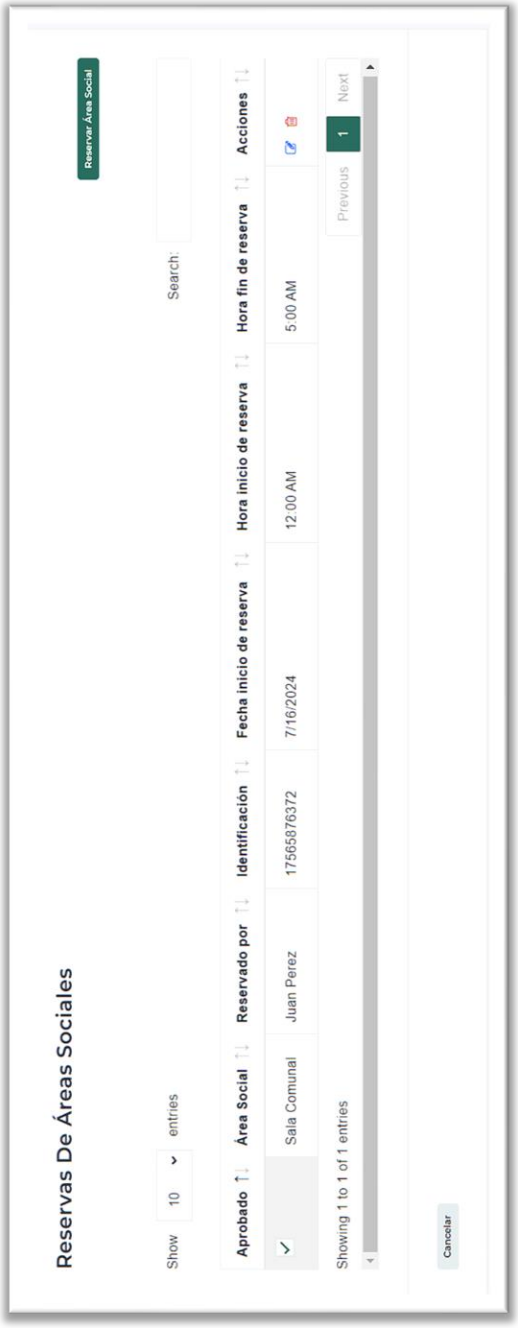


Figura 38. Registro de reservas solicitadas
Elaborado por el autor.

En la figura que se expone a continuación se muestra el registro de las visitas al condominio. En él se plasma el nombre del visitante, el número de ellos, su número de identificación u otros datos. Igualmente, refleja la unidad objeto de visita, la fecha y hora de llegada al igual que la de salida.

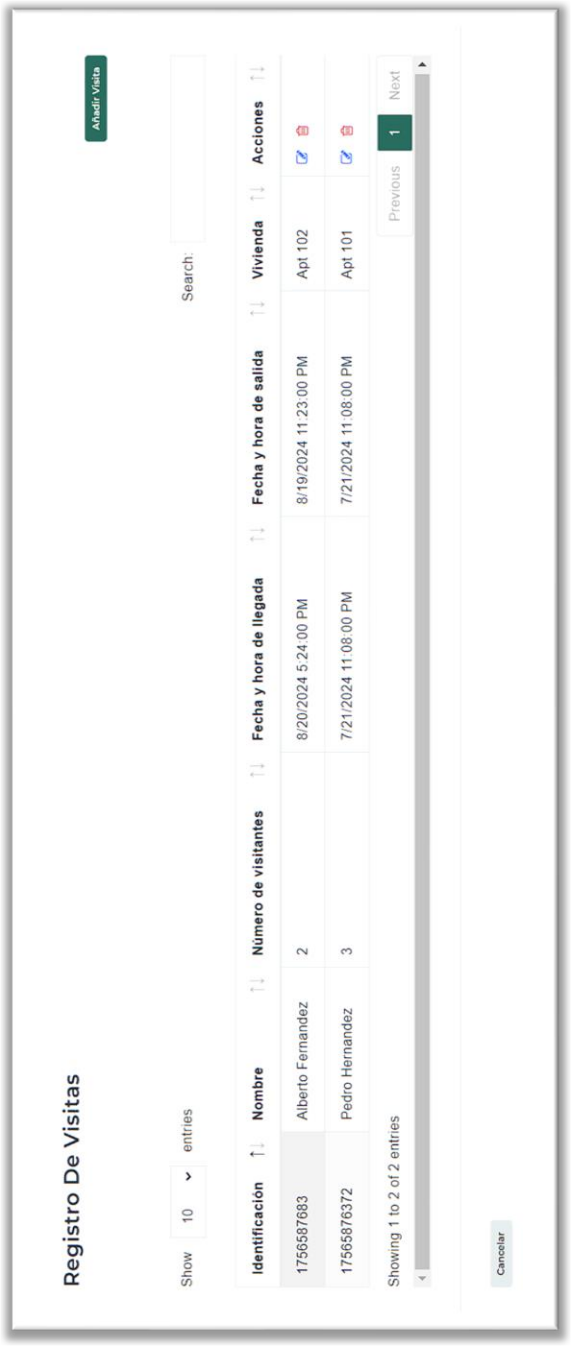


Figura 39. Registro de visitas al condominio.
Elaborado por el autor.

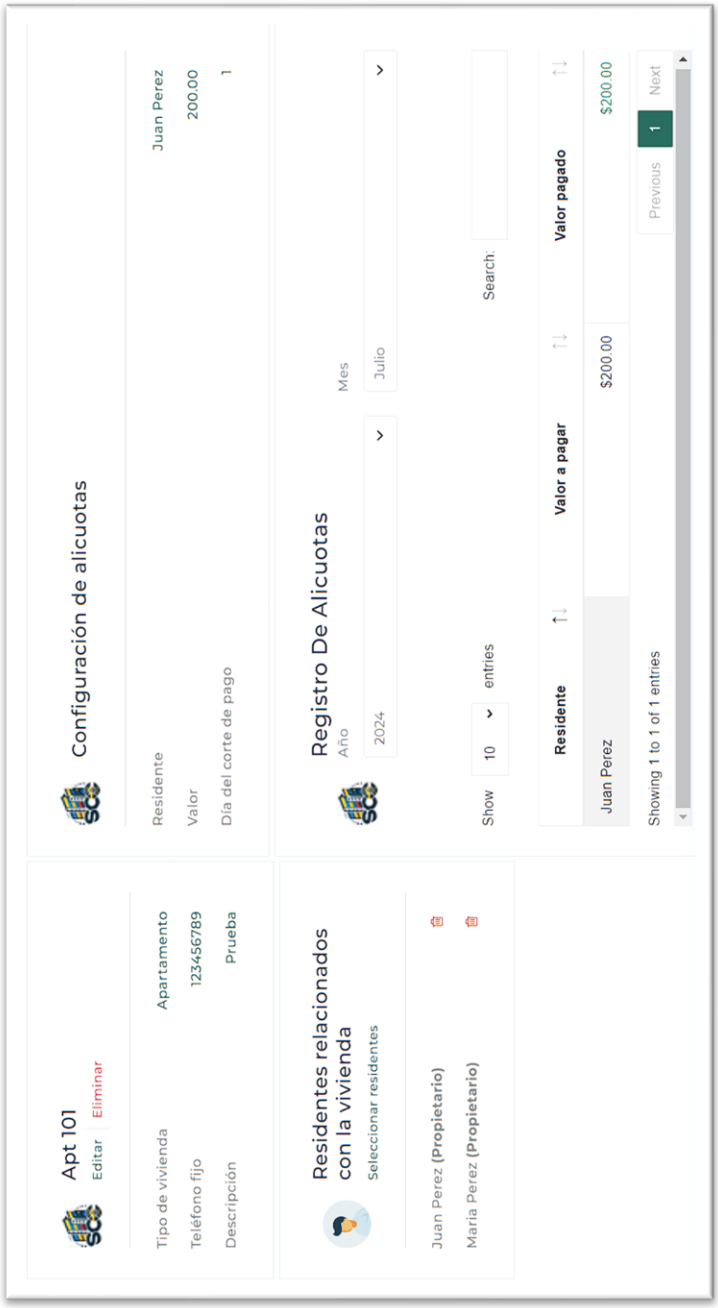


Figura 40. Información sobre la unidad seleccionada.
Elaborado por el autor.

En la figura anterior, se puede observar la información detallada de la unidad seleccionada con su nombre y otros datos. En especial, los usuarios que viven en dicha unidad y la configuración de alícuota que presentan.

En la figura que sigue se aprecian los condominios eliminados anteriormente, teniendo la posibilidad de recuperarlos con todo lo que tenían configurados, para que en caso de que se elimine alguno por error poder recuperarlo con suma facilidad

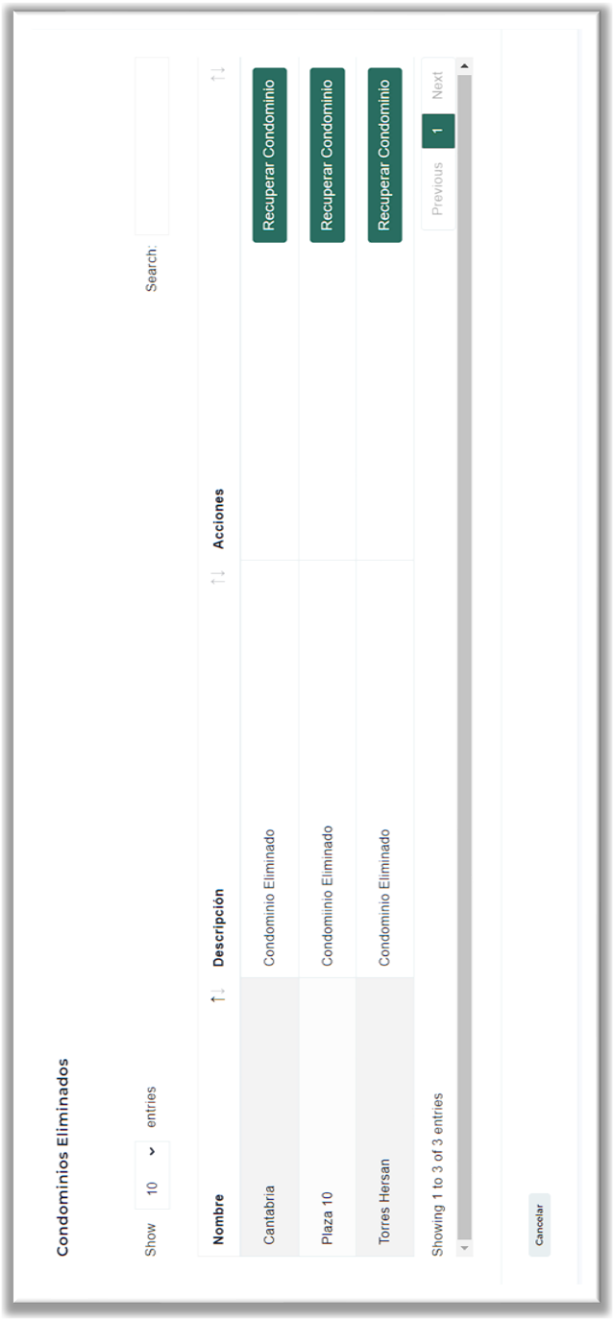


Figura 41. Función para recuperar condominios que han sido eliminados
Elaborado por el autor.

Como se aprecia se muestra, en las figuras mostradas, están los detalles acerca del sistema de control de condominio para el edificio “Plaza 10” con base a la metodología MVC, lo que

se corrobora en la puesta en práctica del sistema que forma parte de este trabajo de investigación.

6.4. Pruebas de verificación del sistema

Corresponde mencionar que el cuarto y último objetivo es el siguiente:

- Realizar las pruebas necesarias del sistema para verificar su correcto funcionamiento.

El objetivo antes expuesto fue cumplido. Para ello y con el fin de verificar el correcto funcionamiento del sistema SCC aplicable al edificio Plaza 10, se siguieron varios pasos en el proceso de pruebas.

En primer lugar, se llevaron a cabo pruebas de integración para evaluar la interacción entre los diferentes componentes del sistema, por ejemplo, el registro de usuarios, registro de reservas, de visitas, etcétera. Ello contribuyó a identificar errores de integración, lo que condujo a la detección de posibles problemas como incompatibilidades de interfaces, errores de comunicación o comportamientos inesperados que surgieron al unir los distintos elementos del sistema y que fueron resueltos para asegurar la eficiencia y funcionalidad optima del sistema.

Igualmente, las pruebas antes mencionadas, permitió asegurar que los componentes se comuniquen correctamente y que los datos fluyan adecuadamente entre ellos. Esto incluyó pruebas de API, servicios y bases de datos para validar la integridad de la información y la coherencia de las operaciones.

Posteriormente, se realizaron pruebas de aceptación o funcionales, donde se simularon escenarios reales de uso del sistema por parte de los usuarios del condominio. Estas se enfocaron en simular la experiencia del usuario, la navegación, la funcionalidad de las

interfaces gráficas y la respuesta del sistema ante acciones comunes. Finalmente, se validó la interacción entre componentes, garantizando que el sistema funcione de manera fluida y sin fallos significativos.

7. Conclusiones y Recomendaciones

7.1. Conclusiones

Se concluye con base a los objetivos trazados lo siguiente:

- El primer objetivo específico se logró mediante un estudio teórico sustentado en que el condominio, específicamente el ubicado en el edificio Plaza 10 implica la coexistencia de bienes comunes que exige su organización al igual que su adecuada y gestión este trabajo demuestra la necesidad de aplicar la tecnología en ellos mediante el desarrollo de un sistema destinado a estos fines. Asimismo, se demuestra teóricamente, que el desarrollo web constituye el proceso de construcción, mantenimiento y actualización de sitios web y aplicaciones que incluye su diseño visual, funcionalidad, interactividad, accesibilidad y optimización para diferentes dispositivos y navegadores. Igualmente, que los lenguajes de programación, entre ellos los aplicados en este estudio, se conforman por conjuntos de instrucciones y reglas que permiten comunicarse con los dispositivos y crear software aplicable a cierto contexto a partir de sus necesidades.

- Como conclusión al segundo objetivo específico de este estudio, se demuestra, mediante una encuesta aplicada a 163 personas que viven en el condominio Plaza 10, que, entre los problemas identificados en este inmueble, están el retraso en los pagos, la falta de comunicación efectiva y la seguridad insuficiente. Por esto, estas áreas críticas requieren una solución integral a través de la implementación de un sistema de condominios que mejore la gestión financiera, la comunicación interna y externa, y refuerce las medidas de seguridad, incluyendo el registro de controles de visitantes y otras herramientas.

- Por otro lado, la encuesta demuestra que, las soluciones y aspectos que debe contener el sistema incluyen una plataforma que permita una gestión financiera eficiente, con seguimiento de pagos y cobros automáticos para evitar retrasos. Además, se necesita un sistema de comunicación interna y externa que vaya más allá de WhatsApp, integrando funciones de mensajería instantánea, notificaciones automáticas y acceso ágil desde dispositivos móviles. Asimismo, para abordar la dificultad en la reserva de áreas comunes, el sistema debe ofrecer un proceso organizado y eficiente para gestionar estas reservas, con disponibilidad en tiempo real y confirmación automática. Asimismo, la interfaz del sistema debe ser moderna, clara y visualmente atractiva, facilitando la interacción entre los residentes y la administración del condominio. Igualmente, se debe garantizar la seguridad mediante el registro y control de visitantes, el monitoreo de áreas comunes a través de cámaras de seguridad y la implementación de medidas para prevenir accesos no autorizados. Todo ello mejorará la convivencia y la calidad de vida en el condominio, cumpliendo con las necesidades identificadas por los residentes.
- En cuanto al objetivo específico tres, se concluye que es necesario desarrollar un sistema de control de condominio para el edificio “Plaza 10” con base a la metodología MVC. Por ello, el sistema de condominios aplica el desarrollo web mediante ASP.NET. Core y el gestor de base de datos SQL Server en cuanto al backend. Por su lado, el frontend se empleó HTML5, CSS3, JavaScript y ASP.NET. Core. Entre las principales funcionalidades del sistema, creadas para resolver las problemáticas de administración actuales existentes en el edificio están: la creación del condominio con sus respectivas áreas, unidades y usuarios. También, la implementación de registro de visitantes, reservas de áreas comunes como gimnasio, BBQ, sala comunal. De igual modo, la creación de directivas según sus periodos, la configuración de alícuotas por unidades y el registro del pago, de estas, por mes y año, entre otras.

- El cumplimiento del cuarto objetivo específico que está enfocado en verificar el correcto funcionamiento del sistema SCC para el edificio Plaza 10, se logró mediante un proceso de pruebas. Entre ellas, se aplicaron las de integración que permitieron detectar y resolver errores de interacción entre los componentes del sistema, asegurando así su eficiencia y funcionalidad óptima. Asimismo, las pruebas de comunicación entre componentes y las pruebas funcionales garantizaron la integridad de los datos y la experiencia satisfactoria del usuario en situaciones reales de uso. Este proceso validó la interacción fluida y sin fallos significativos entre los distintos elementos del sistema SCC. Como resultado, se obtiene una herramienta tecnológica efectiva que contribuye a mejorar la gestión del condominio Plaza 10, bajo condiciones de eficiencia, calidad y funcionalidad.
- Por lo anterior se afirma que el objetivo general formulado en esta investigación fue cumplido cabalmente, puesto que se desarrolló e implementó un sistema integral de control de condominio para el edificio Plaza 10 enfocado a mejorar la gestión administrativa, reservas de áreas comunes, consulta de pagos de alícuotas, registro de visitas, entre otras funciones para lograr la eficiencia en la administración y mejorar la calidad de vida de sus habitantes.

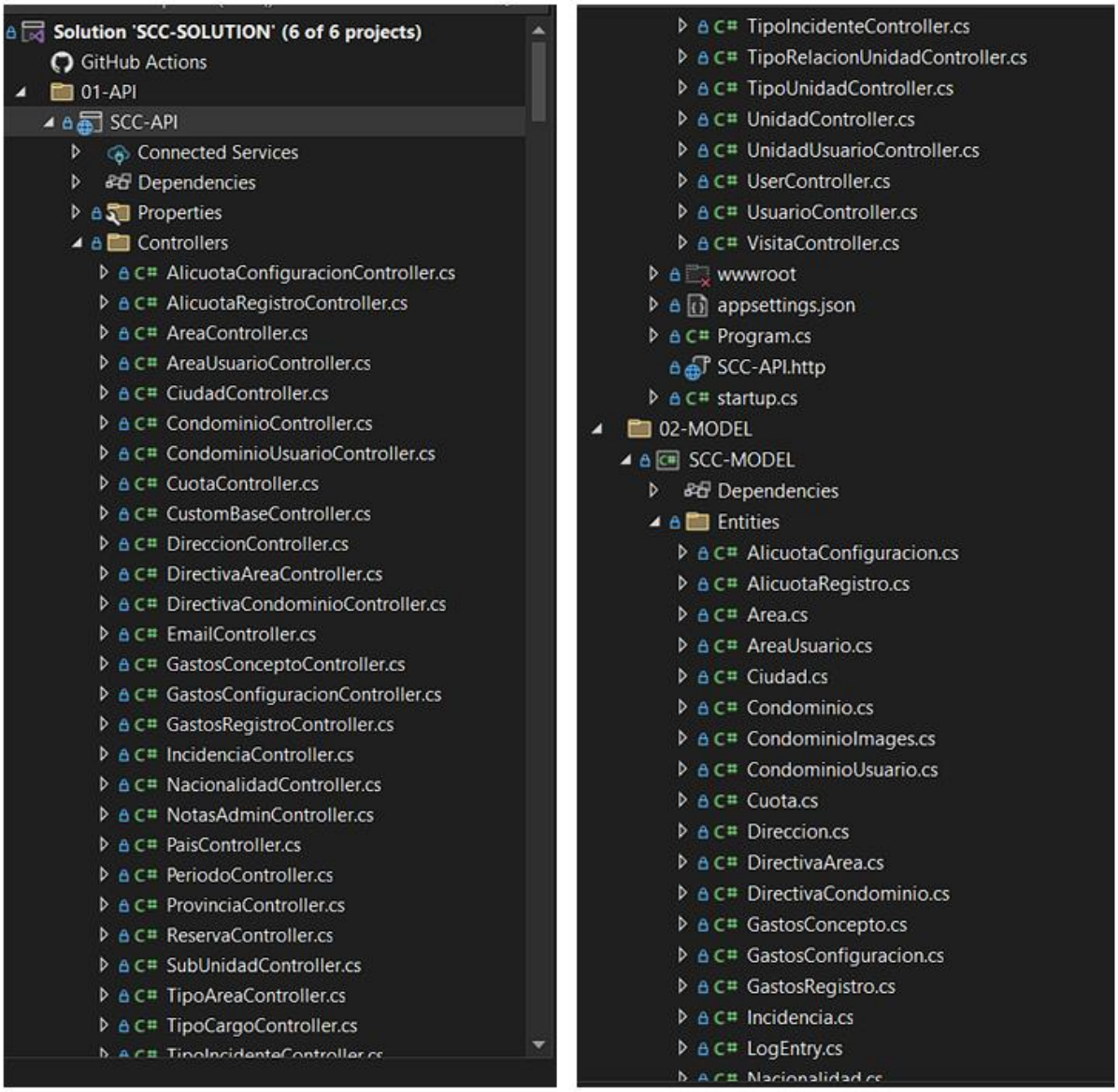
7.2. Recomendaciones

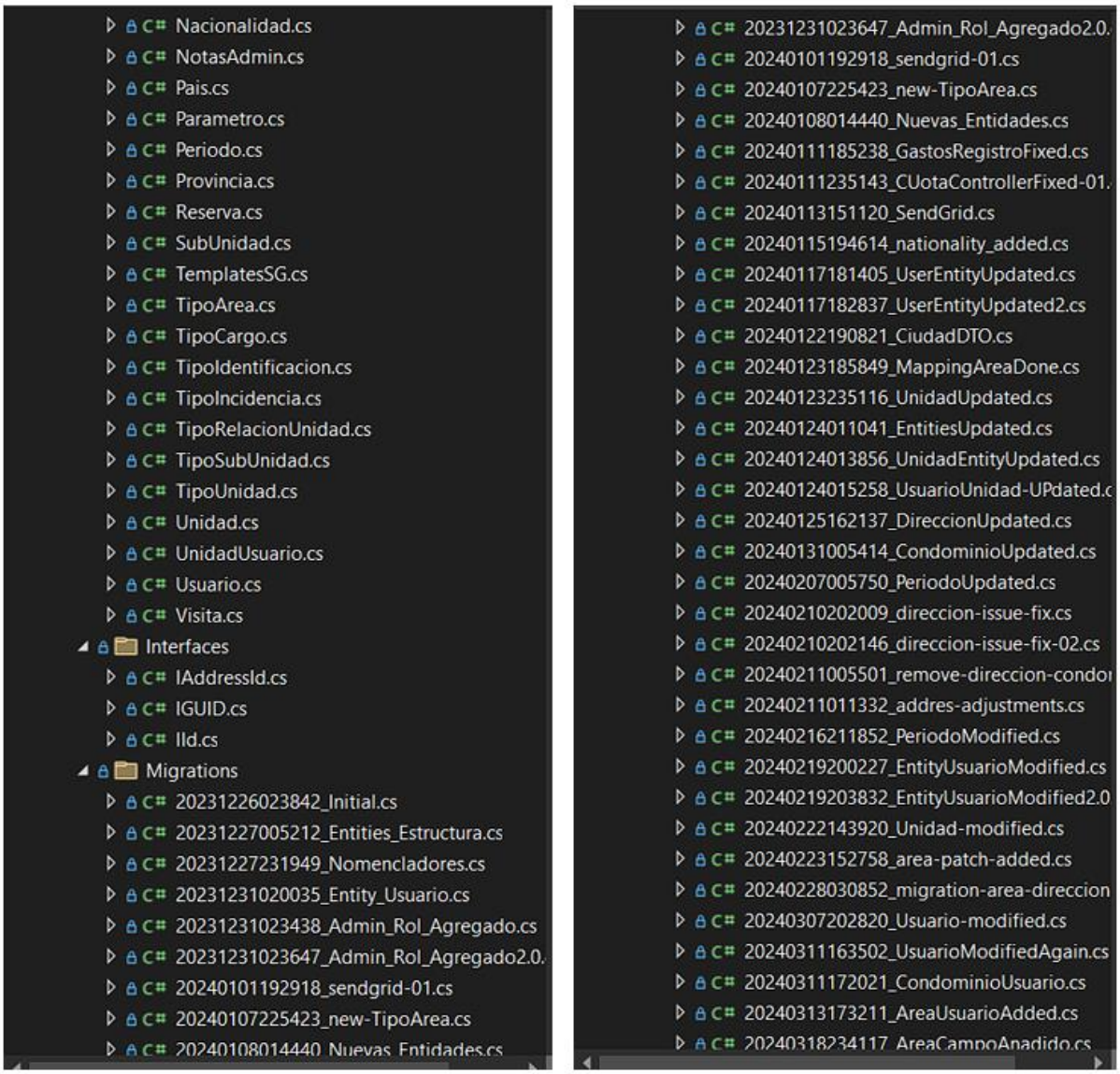
Con base al estudio realizado, se recomienda lo siguiente:

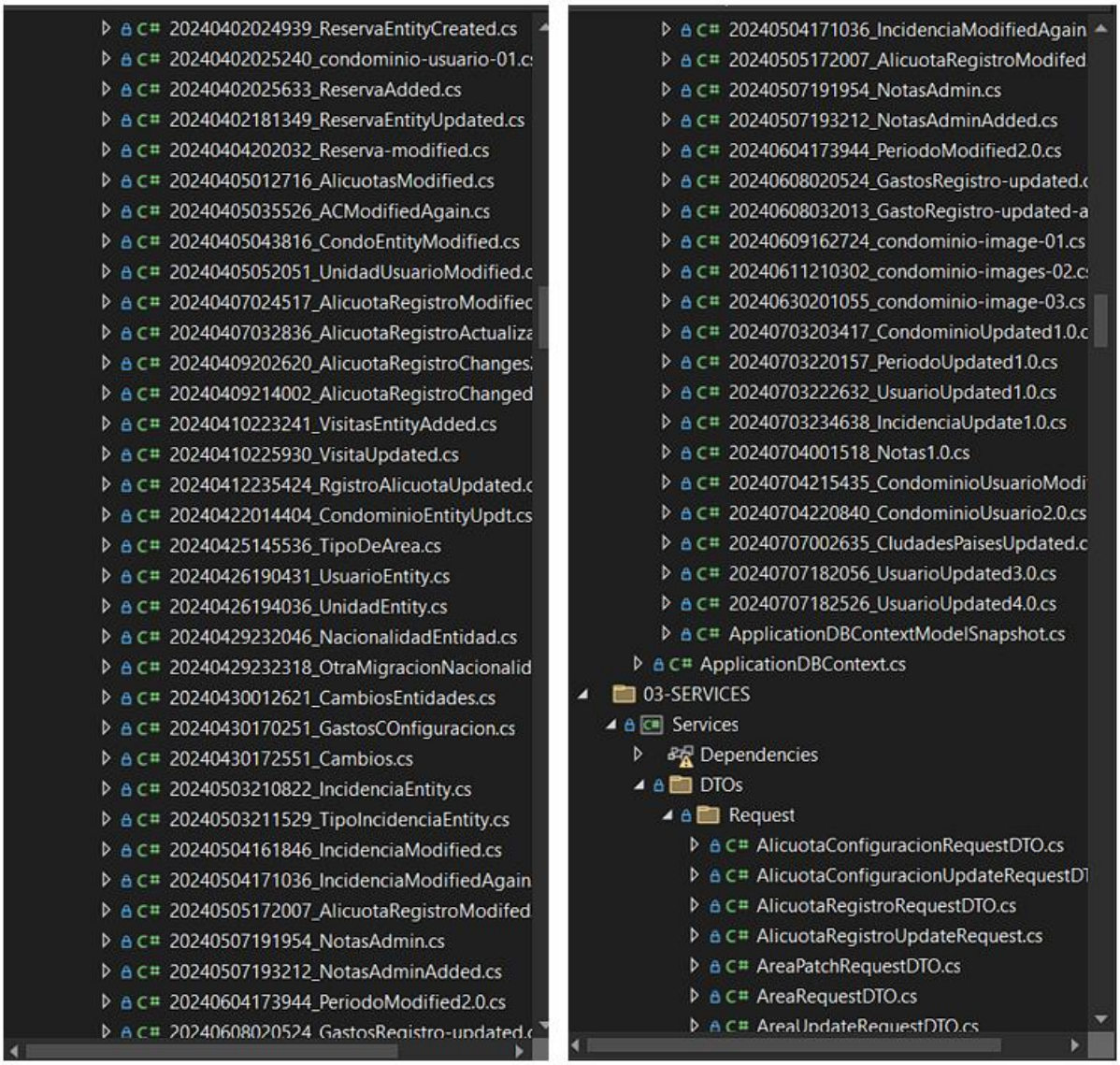
- Se recomienda, para asegurar que el sistema de condominio gane mayor integralidad se sugiere crear una interfaz de usuario para que el sistema no solo funcione para los administradores, sino también para los residentes.
- Se sugiere implementar un sistema de mensajería instantánea una vez creada la interfaz para los residentes.
- Se sugiere a futuro crear una aplicación móvil para que pueda ser utilizada desde dispositivos más cómodos como son las tablets o teléfonos celulares.
- Se recomienda con la finalidad de ampliar el sistema se sugiere que, a futuro, se agregue la funcionalidad de un registro de gastos mensuales del condominio.
- Se recomienda gestionar la publicidad de este sistema con la finalidad de comercializarlo entre el mercado inmobiliario para de esta forma aplicarlo y asegurar el éxito en la gestión de los condominios.
- Se sugiere integrar un sistema de pagos para que cada mes el propietario de la vivienda pueda cancelar el valor mensual de la alícuota.

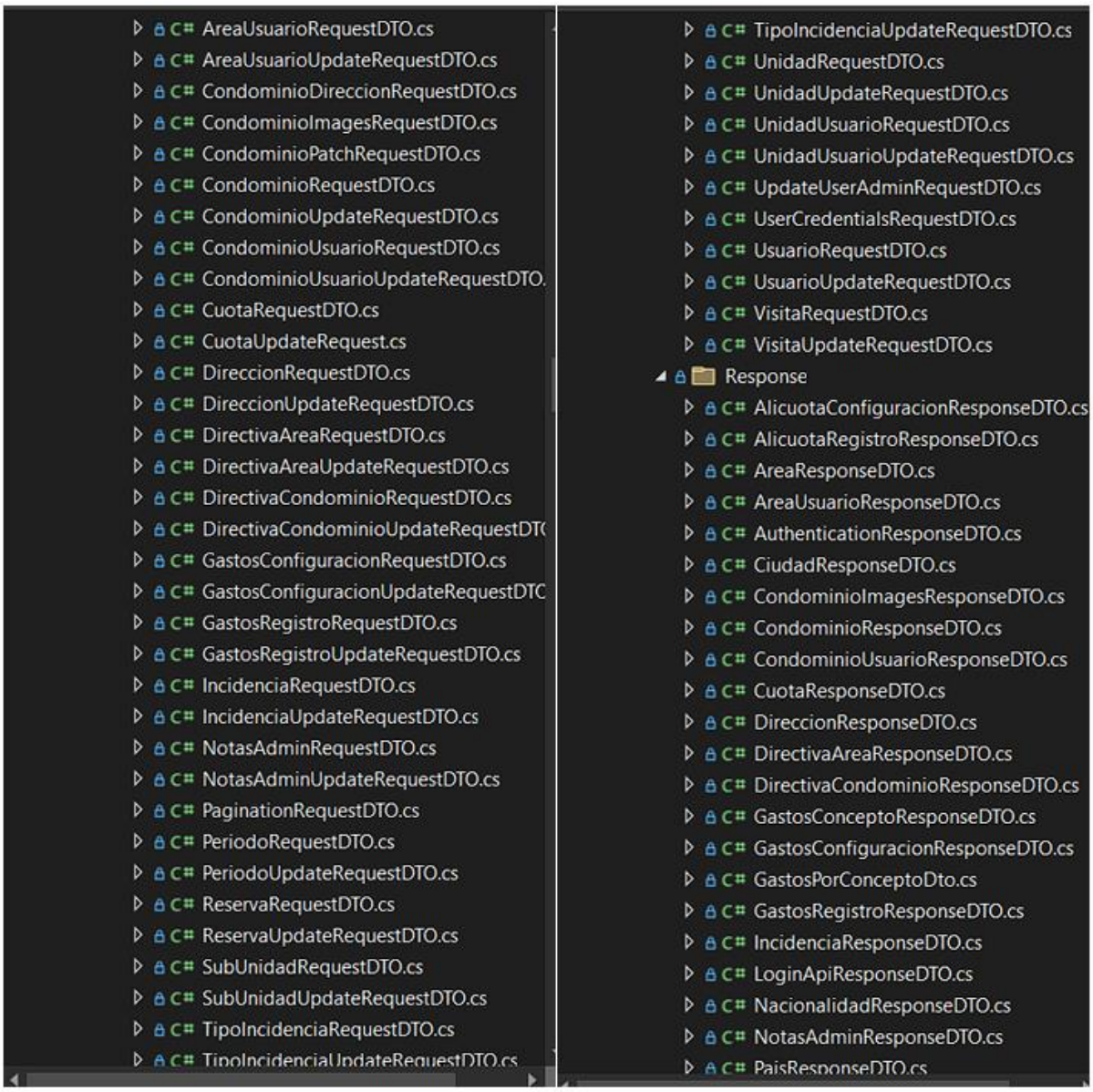
Evidencia del desarrollo del sistema

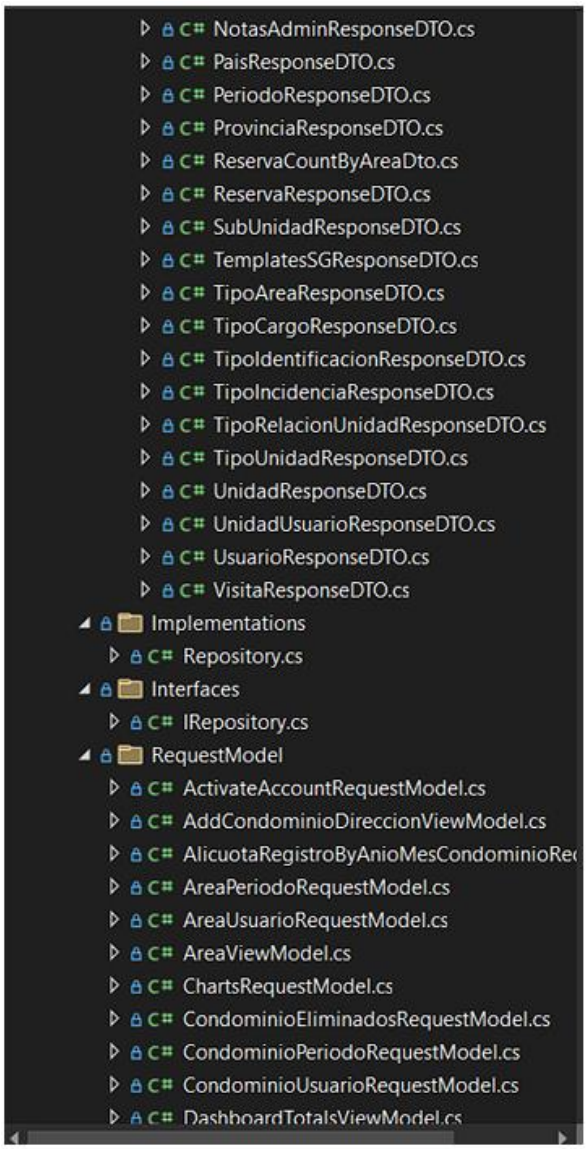
A continuación, se muestran las capturas de pantalla de todos los archivos que constan dentro del sistema para su respectiva evidencia

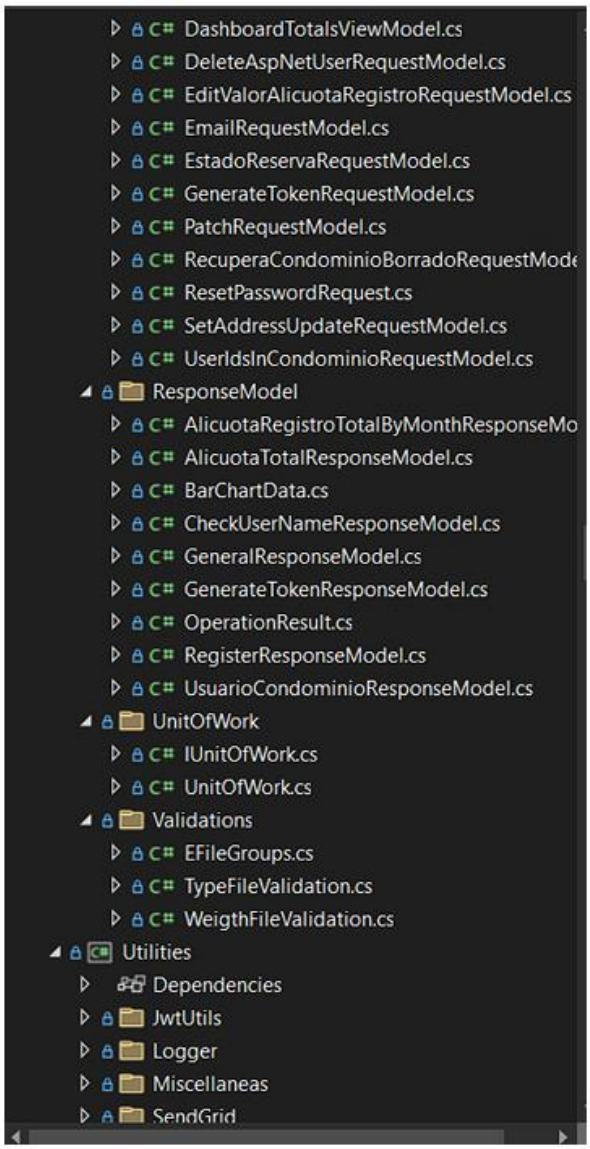



Nacionalidad.cs
NotasAdmin.cs
Pais.cs
Parametro.cs
Periodo.cs
Provincia.cs
Reserva.cs
SubUnidad.cs
TemplatesSG.cs
TipoArea.cs
TipoCargo.cs
TipoIdentificacion.cs
TipoIncidencia.cs
TipoRelacionUnidad.cs
TipoSubUnidad.cs
TipoUnidad.cs
Unidad.cs
UnidadUsuario.cs
Usuario.cs
Visita.cs
Interfaces
IAddressId.cs
IGUID.cs
IId.cs
Migrations
20231226023842_Initial.cs
20231227005212_Entities_Estructura.cs
20231227231949_Nomencladores.cs
20231231020035_Entity_Usuario.cs
20231231023438_Admin_Rol_Agregado.cs
20231231023647_Admin_Rol_Agregado2.0.
20240101192918_sendgrid-01.cs
20240107225423_new-TipoArea.cs
20240108014440_Nuevas_Entidades.cs
20231231023647_Admin_Rol_Agregado2.0.
20240101192918_sendgrid-01.cs
20240107225423_new-TipoArea.cs
20240108014440_Nuevas_Entidades.cs
20240111185238_GastosRegistroFixed.cs
20240111235143_CUotaControllerFixed-01.
20240113151120_SendGrid.cs
20240115194614_nationality_added.cs
20240117181405_UserEntityUpdated.cs
20240117182837_UserEntityUpdated2.cs
20240122190821_CiudadDTO.cs
20240123185849_MappingAreaDone.cs
20240123235116_UnidadUpdated.cs
20240124011041_EntitiesUpdated.cs
20240124013856_UnidadEntityUpdated.cs
20240124015258_UsuarioUnidad-UPdated.c
20240125162137_DireccionUpdated.cs
20240131005414_CondominioUpdated.cs
20240207005750_PeriodoUpdated.cs
20240210202009_direccion-issue-fix.cs
20240210202146_direccion-issue-fix-02.cs
20240211005501_remove-direccion-condo
20240211011332_addres-adjustments.cs
20240216211852_PeriodoModified.cs
20240219200227_EntityUsuarioModified.cs
20240219203832_EntityUsuarioModified2.0
20240222143920_Unidad-modified.cs
20240223152758_area-patch-added.cs
20240228030852_migration-area-direccion
20240307202820_Usuario-modified.cs
20240311163502_UsuarioModifiedAgain.cs
20240311172021_CondominioUsuario.cs
20240313173211_AreaUsuarioAdded.cs
20240318234117_AreaCampoAnadido.cs

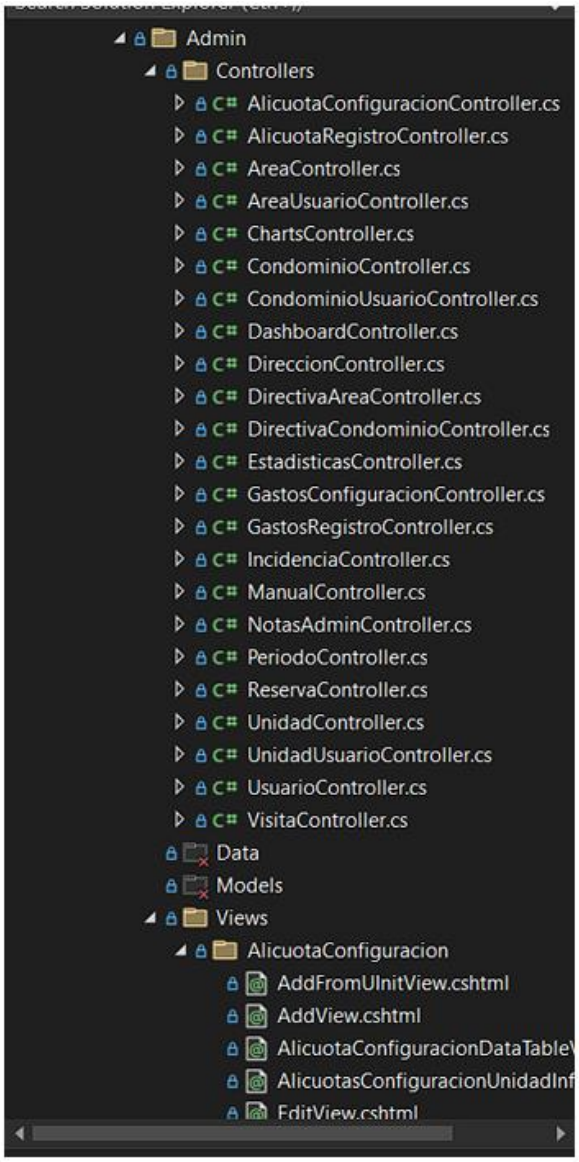

20240402024939_ReservaEntityCreated.cs
20240402025240_condominio-usuario-01.cs
20240402025633_ReservaAdded.cs
20240402181349_ReservaEntityUpdated.cs
20240404202032_Reserva-modified.cs
20240405012716_AlicuotasModified.cs
20240405035526_ACModifiedAgain.cs
20240405043816_CondoEntityModified.cs
20240405052051_UnidadUsuarioModified.c
20240407024517_AlicuotaRegistroModifiec
20240407032836_AlicuotaRegistroActualiza
20240409202620_AlicuotaRegistroChanges
20240409214002_AlicuotaRegistroChanged
20240410223241_VisitasEntityAdded.cs
20240410225930_VisitaUpdated.cs
20240412235424_RgistroAlicuotaUpdated.c
20240422014404_CondominioEntityUpdt.cs
20240425145536_TipoDeArea.cs
20240426190431_UsuarioEntity.cs
20240426194036_UnidadEntity.cs
20240429232046_NacionalidadEntidad.cs
20240429232318_OtraMigracionNacionalid
20240430012621_CambiosEntidades.cs
20240430170251_GastosCOnfiguracion.cs
20240430172551_Cambios.cs
20240503210822_IncidenciaEntity.cs
20240503211529_TipoIncidenciaEntity.cs
20240504161846_IncidenciaModified.cs
20240504171036_IncidenciaModifiedAgain
20240505172007_AlicuotaRegistroModifed
20240507191954_NotasAdmin.cs
20240507193212_NotasAdminAdded.cs
20240604173944_PeriodoModified2.0.cs
20240608020524_GastosRegistro-updated.c
20240504171036_IncidenciaModifiedAgain
20240505172007_AlicuotaRegistroModifed
20240507191954_NotasAdmin.cs
20240507193212_NotasAdminAdded.cs
20240604173944_PeriodoModified2.0.cs
20240608020524_GastosRegistro-updated.c
20240608032013_GastoRegistro-updated-a
20240609162724_condominio-image-01.cs
20240611210302_condominio-images-02.cs
20240630201055_condominio-image-03.cs
20240703203417_CondominioUpdated1.0.c
20240703220157_PeriodoUpdated1.0.cs
20240703222632_UsuarioUpdated1.0.cs
20240703234638_IncidenciaUpdate1.0.cs
20240704001518_Notas1.0.cs
20240704215435_CondominioUsuarioModi
20240704220840_CondominioUsuario2.0.cs
20240707002635_CludadesPaisesUpdated.c
20240707182056_UsuarioUpdated3.0.cs
20240707182526_UsuarioUpdated4.0.cs
ApplicationDBContextModelSnapshot.cs
ApplicationDBContext.cs
03-SERVICES
Services
Dependencies
DTOs
Request
AlicuotaConfiguracionRequestDTO.cs
AlicuotaConfiguracionUpdateRequestDT
AlicuotaRegistroRequestDTO.cs
AlicuotaRegistroUpdateRequest.cs
AreaPatchRequestDTO.cs
AreaRequestDTO.cs
AreaUpdateRequestDTO.cs

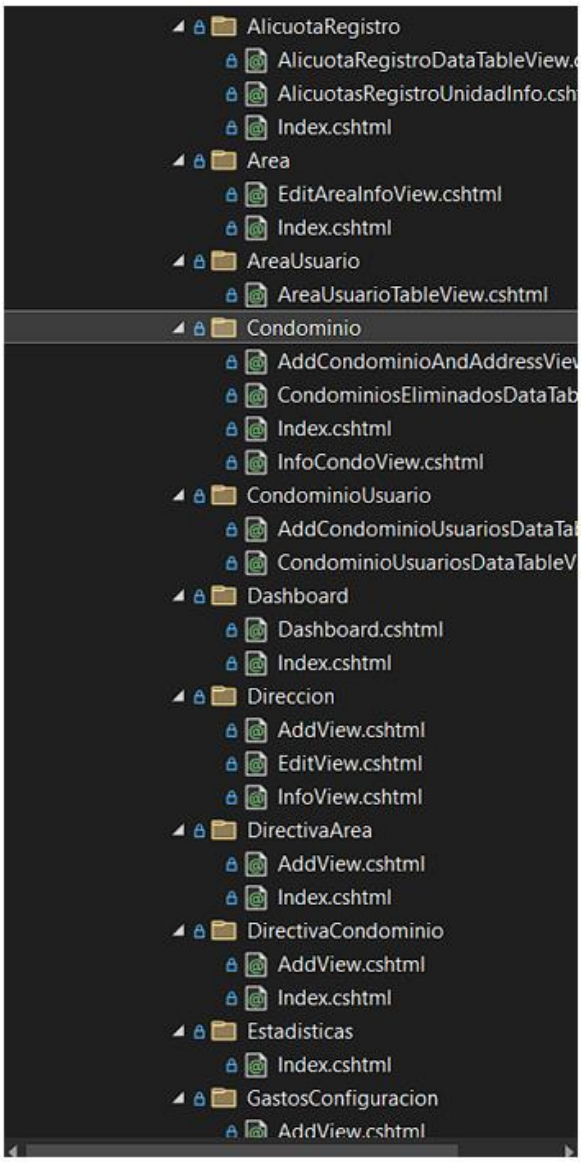

AreaUsuarioRequestDTO.cs
AreaUsuarioUpdateRequestDTO.cs
CondominioDireccionRequestDTO.cs
CondominioImagesRequestDTO.cs
CondominioPatchRequestDTO.cs
CondominioRequestDTO.cs
CondominioUpdateRequestDTO.cs
CondominioUsuarioRequestDTO.cs
CondominioUsuarioUpdateRequestDTO.
CuotaRequestDTO.cs
CuotaUpdateRequest.cs
DireccionRequestDTO.cs
DireccionUpdateRequestDTO.cs
DirectivaAreaRequestDTO.cs
DirectivaAreaUpdateRequestDTO.cs
DirectivaCondominioRequestDTO.cs
DirectivaCondominioUpdateRequestDT
GastosConfiguracionRequestDTO.cs
GastosConfiguracionUpdateRequestDTC
GastosRegistroRequestDTO.cs
GastosRegistroUpdateRequestDTO.cs
IncidenciaRequestDTO.cs
IncidenciaUpdateRequestDTO.cs
NotasAdminRequestDTO.cs
NotasAdminUpdateRequestDTO.cs
PaginationRequestDTO.cs
PeriodoRequestDTO.cs
PeriodoUpdateRequestDTO.cs
ReservaRequestDTO.cs
ReservaUpdateRequestDTO.cs
SubUnidadRequestDTO.cs
SubUnidadUpdateRequestDTO.cs
TipoIncidenciaRequestDTO.cs
TipoIncidenciaUpdateRequestDTO.cs
UnidadRequestDTO.cs
UnidadUpdateRequestDTO.cs
UnidadUsuarioRequestDTO.cs
UnidadUsuarioUpdateRequestDTO.cs
UpdateUserAdminRequestDTO.cs
UserCredentialsRequestDTO.cs
UsuarioRequestDTO.cs
UsuarioUpdateRequestDTO.cs
VisitaRequestDTO.cs
VisitaUpdateRequestDTO.cs
Response
AlicuotaConfiguracionResponseDTO.cs
AlicuotaRegistroResponseDTO.cs
AreaResponseDTO.cs
AreaUsuarioResponseDTO.cs
AuthenticationResponseDTO.cs
CiudadResponseDTO.cs
CondominioImagesResponseDTO.cs
CondominioResponseDTO.cs
CondominioUsuarioResponseDTO.cs
CuotaResponseDTO.cs
DireccionResponseDTO.cs
DirectivaAreaResponseDTO.cs
DirectivaCondominioResponseDTO.cs
GastosConceptoResponseDTO.cs
GastosConfiguracionResponseDTO.cs
GastosPorConceptoDto.cs
GastosRegistroResponseDTO.cs
IncidenciaResponseDTO.cs
LoginApiResponseDTO.cs
NacionalidadResponseDTO.cs
NotasAdminResponseDTO.cs
PaisResponseDTO.cs

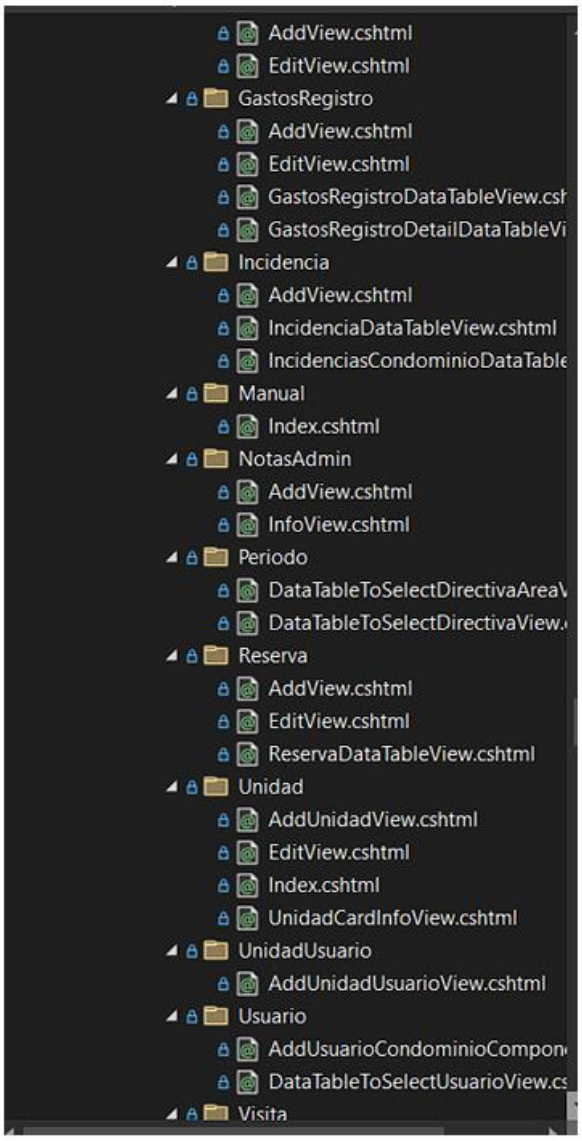

NotasAdminResponseDTO.cs
PaisResponseDTO.cs
PeriodoResponseDTO.cs
ProvinciaResponseDTO.cs
ReservaCountByAreaDto.cs
ReservaResponseDTO.cs
SubUnidadResponseDTO.cs
TemplatesSGResponseDTO.cs
TipoAreaResponseDTO.cs
TipoCargoResponseDTO.cs
TipoIdentificacionResponseDTO.cs
TipoIncidenciaResponseDTO.cs
TipoRelacionUnidadResponseDTO.cs
TipoUnidadResponseDTO.cs
UnidadResponseDTO.cs
UnidadUsuarioResponseDTO.cs
UsuarioResponseDTO.cs
VisitaResponseDTO.cs
Implementations
Repository.cs
Interfaces
IRepository.cs
RequestModel
ActivateAccountRequestModel.cs
AddCondominioDireccionViewModel.cs
AreaPeriodoRequestModel.cs
AreaUsuarioRequestModel.cs
AreaViewModel.cs
ChartsRequestModel.cs
CondominioEliminadosRequestModel.cs
CondominioPeriodoRequestModel.cs
CondominioUsuarioRequestModel.cs

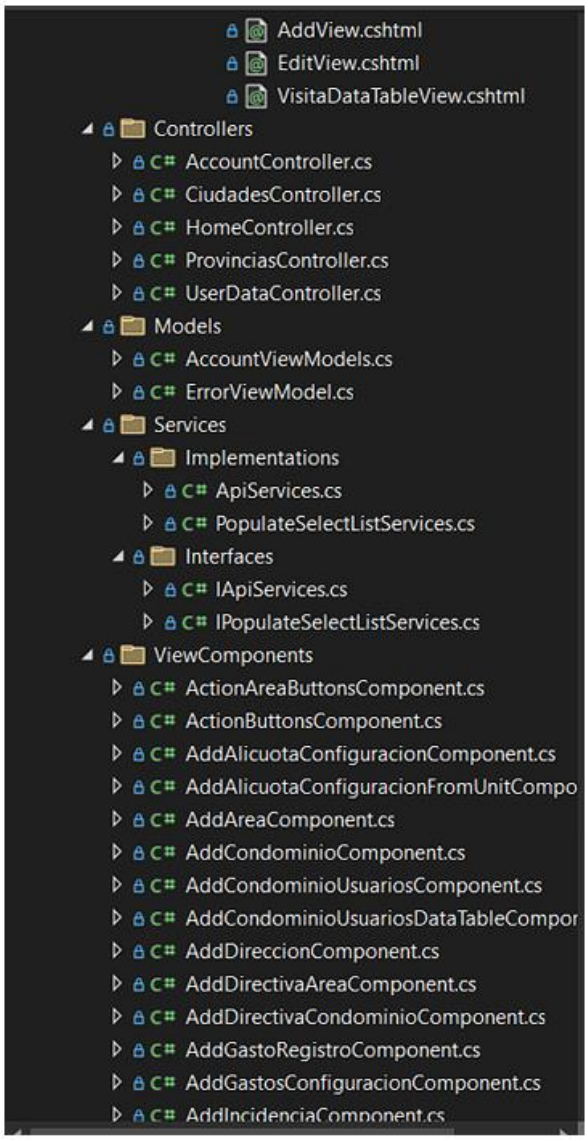

DashboardTotalsViewModel.cs
DeleteAspNetUserRequestModel.cs
EditValorAlicuotaRegistroRequestModel.cs
EmailRequestModel.cs
EstadoReservaRequestModel.cs
GenerateTokenRequestModel.cs
PatchRequestModel.cs
ResetPasswordRequest.cs
SetAddressUpdateRequestModel.cs
UserIdsInCondominioRequestModel.cs
ResponseModel
AlicuotaTotalResponseModel.cs
BarChartData.cs
CheckUserNameResponseModel.cs
GeneralResponseModel.cs
GenerateTokenResponseModel.cs
OperationResult.cs
RegisterResponseModel.cs
UsuarioCondominioResponseModel.cs
UnitOfWork
IUnitOfWork.cs
UnitOfWork.cs
Validations
EFileGroups.cs
TypeFileValidation.cs
WeigthFileValidation.cs
Utilities
Dependencies
JwtUtils
Logger
Miscellaneas

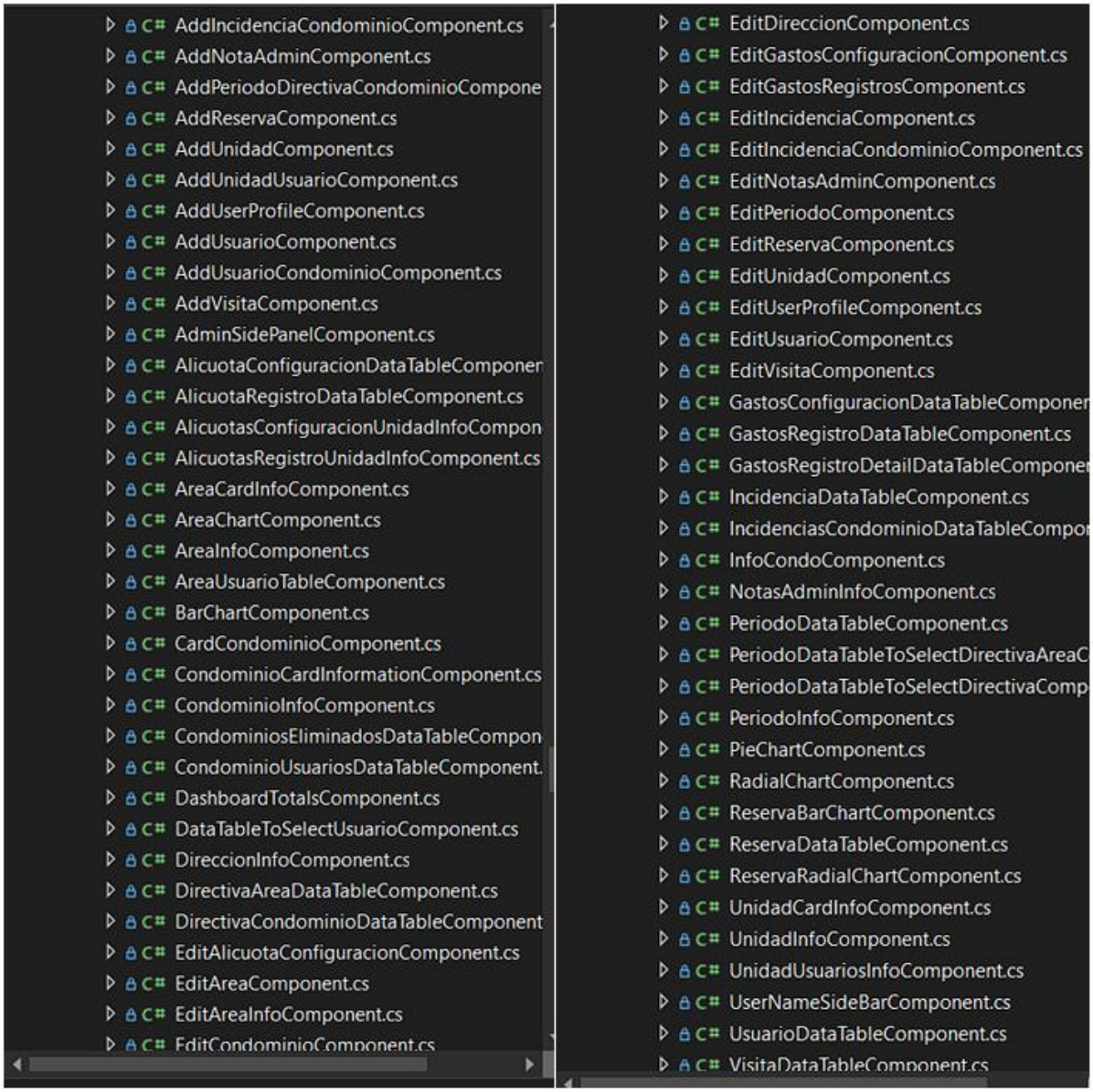

Utilities
Dependencies
JwtUtils
JwtUtils.cs
Logger
DatabaseLogger.cs
DatabaseLoggerProvider.cs
Miscellaneas
Miscellaneas.cs
SendGrid
EmailService.cs
IEmailService.cs
SendGridSettings.cs
AutoMapperProfile.cs
HttpContextExtensions.cs
IQueryableExtensions.cs
04-Common
Common
Dependencies
Resources
FileResource.resx
Constants.cs
ETipoRelacionUnidad.cs
05-FRONTEND
SCC-FRONTEND
Connected Services
Dependencies
Properties
wwwroot
assets
ajax
audio
css
fonts
wwwroot
assets
ajax
audio
css
fonts
images
js
json
pug
scss
video
css
site.css
js
ajax-utils.js
alicuota-configuracion-functions.js
alicuota-registro-functions.js
common-script.js
condo-charts.js
condominios-eliminados-functions.js
condominio-usuario-functions.js
directiva-area-functions.js
directiva-condominio-functions.js
periodo-gasto-configuracion-functions
reserva-functions.js
site.js
unidad-usuario-functions.js
lib
favicon.ico
Areas
Admin
Controllers
AlicuotaConfiguracionController.cs

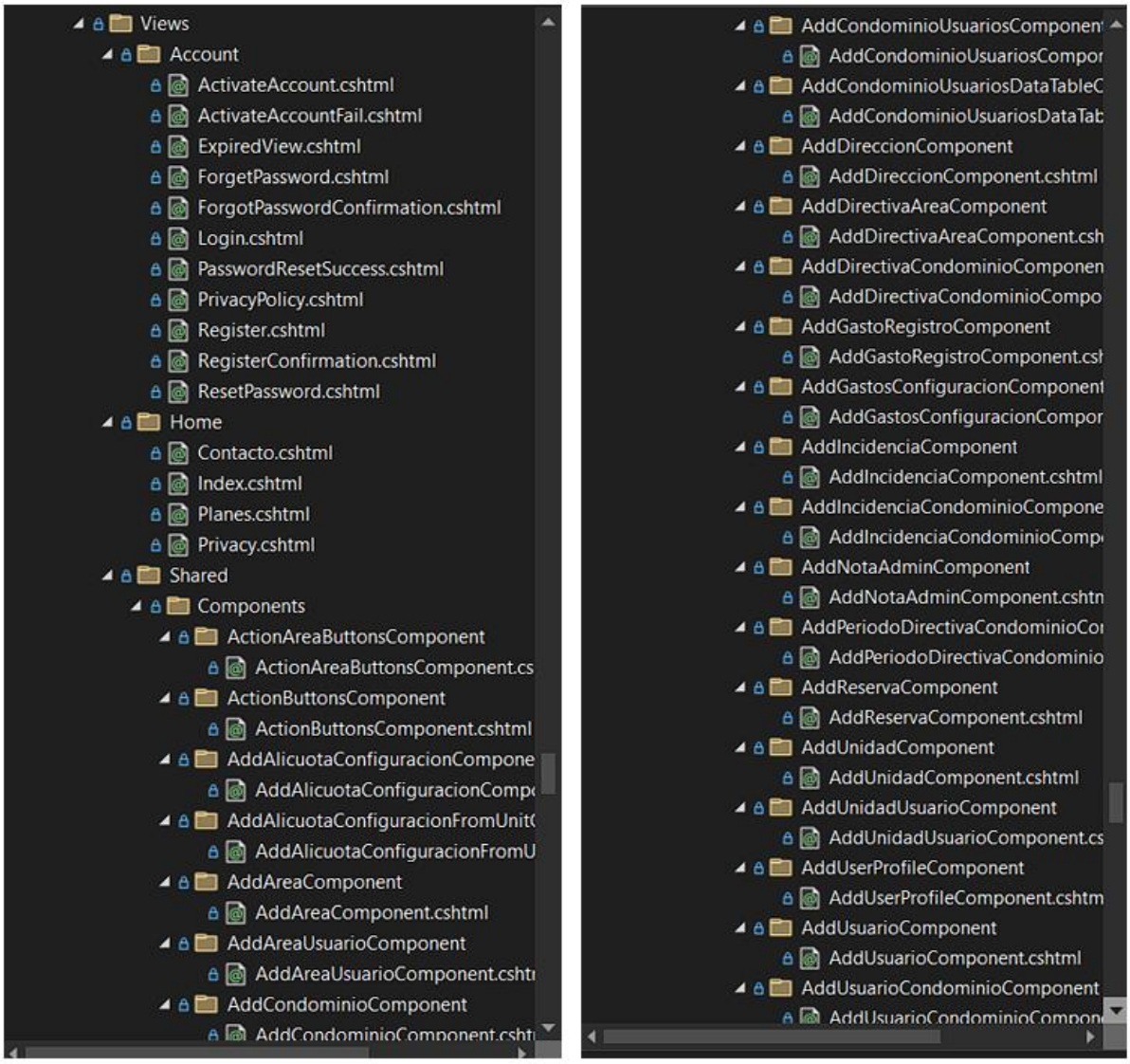

Admin
Controllers
AlicuotaConfiguracionController.cs
AlicuotaRegistroController.cs
AreaController.cs
AreaUsuarioController.cs
ChartsController.cs
CondominioController.cs
CondominioUsuarioController.cs
DashboardController.cs
DireccionController.cs
DirectivaAreaController.cs
DirectivaCondominioController.cs
EstadisticasController.cs
GastosConfiguracionController.cs
GastosRegistroController.cs
IncidenciaController.cs
ManualController.cs
NotasAdminController.cs
PeriodoController.cs
ReservaController.cs
UnidadController.cs
UnidadUsuarioController.cs
UsuarioController.cs
VisitaController.cs
Data
Models
Views
AlicuotaConfiguracion
AddFromUnitView.cshtml
AddView.cshtml
AlicuotasConfiguracionUnidadInf
EditView.cshtml

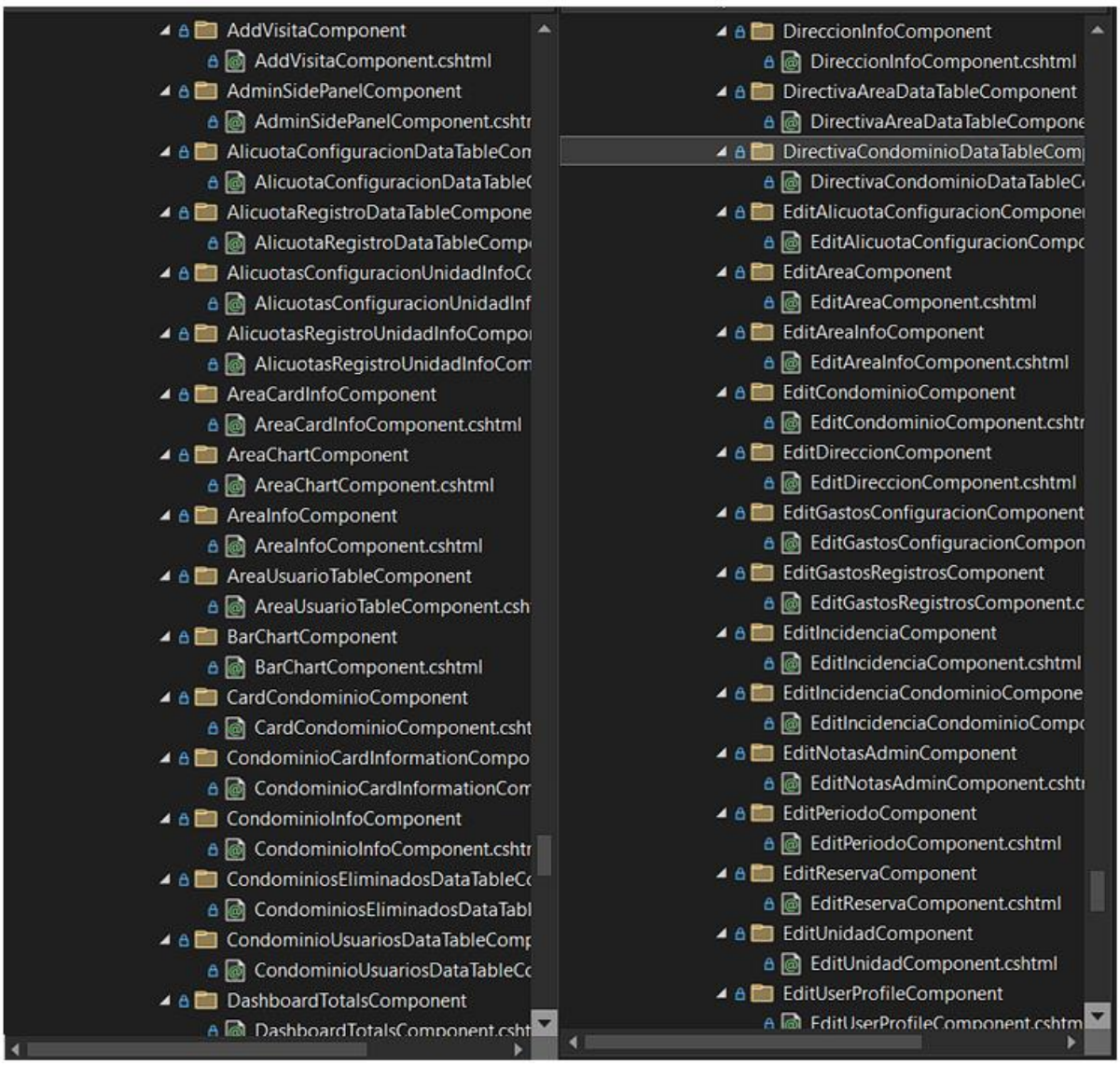

AlicuotaRegistro
AlicuotaRegistroDataTableView.
AlicuotasRegistroUnidadInfo.csh
Index.cshtml
Area
EditAreaInfoView.cshtml
Index.cshtml
AreaUsuario
AreaUsuarioTableView.cshtml
Condominio
AddCondominioAndAddressVie
CondominiosEliminadosDataTab
Index.cshtml
InfoCondoView.cshtml
CondominioUsuario
AddCondominioUsuariosDataTa
CondominioUsuariosDataTableV
Dashboard
Dashboard.cshtml
Index.cshtml
Direccion
AddView.cshtml
EditView.cshtml
InfoView.cshtml
DirectivaArea
AddView.cshtml
Index.cshtml
DirectivaCondominio
AddView.cshtml
Index.cshtml
Estadisticas
Index.cshtml
GastosConfiguracion
AddView.cshtml

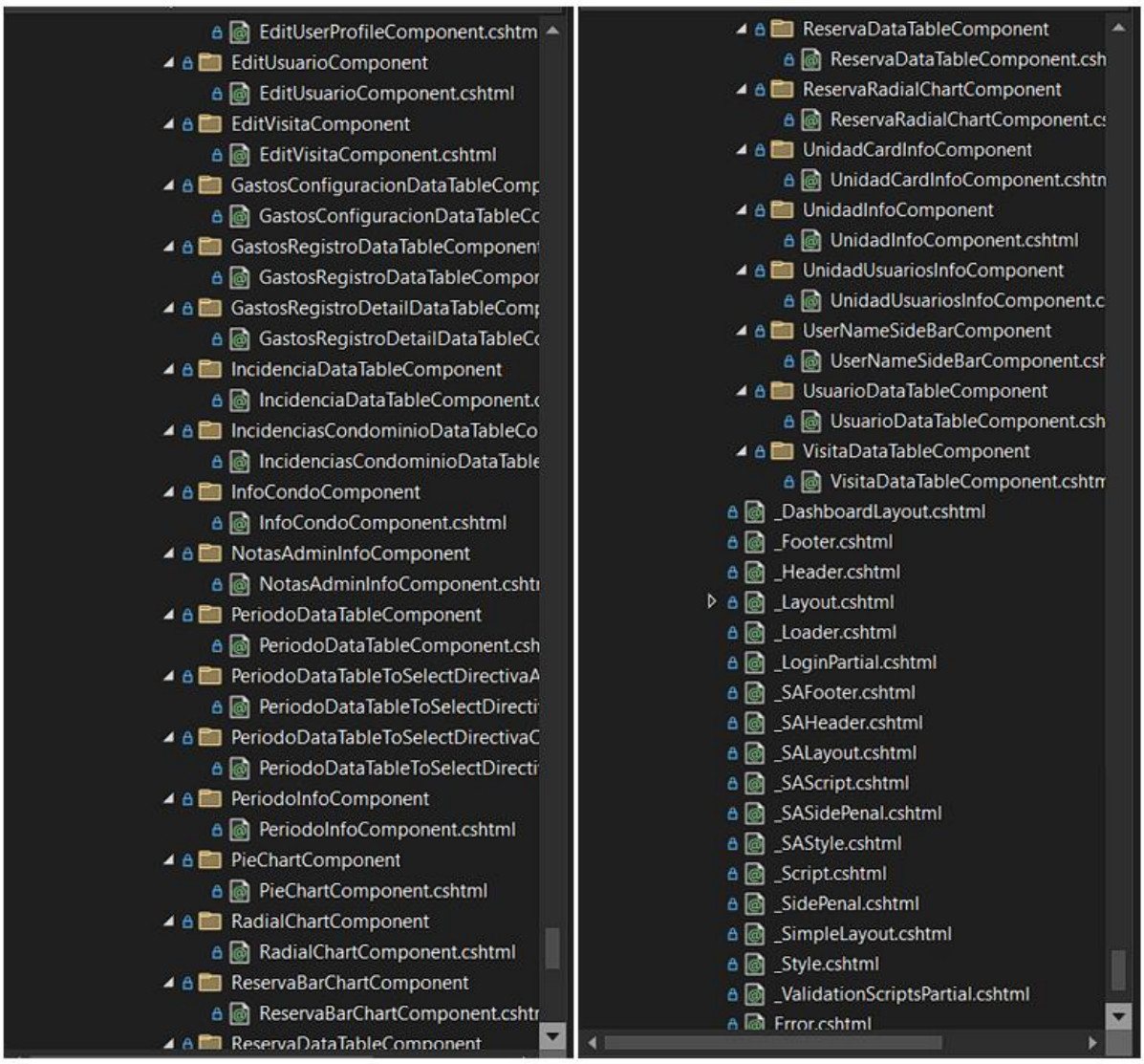

AddView.cshtml
EditView.cshtml
GastosRegistro
AddView.cshtml
EditView.cshtml
GastosRegistroDataTableView.csh
GastosRegistroDetailDataTableVi
Incidencia
AddView.cshtml
IncidenciaDataTableView.cshtml
IncidenciasCondominioDataTable
Manual
Index.cshtml
NotasAdmin
AddView.cshtml
InfoView.cshtml
Periodo
DataTableToSelectDirectivaAreaV
DataTableToSelectDirectivaView.
Reserva
AddView.cshtml
EditView.cshtml
ReservaDataTableView.cshtml
Unidad
AddUnidadView.cshtml
EditView.cshtml
Index.cshtml
UnidadCardInfoView.cshtml
UnidadUsuario
AddUnidadUsuarioView.cshtml
Usuario
AddUsuarioCondominioCompon
DataTableToSelectUsuarioView.cs
Visita

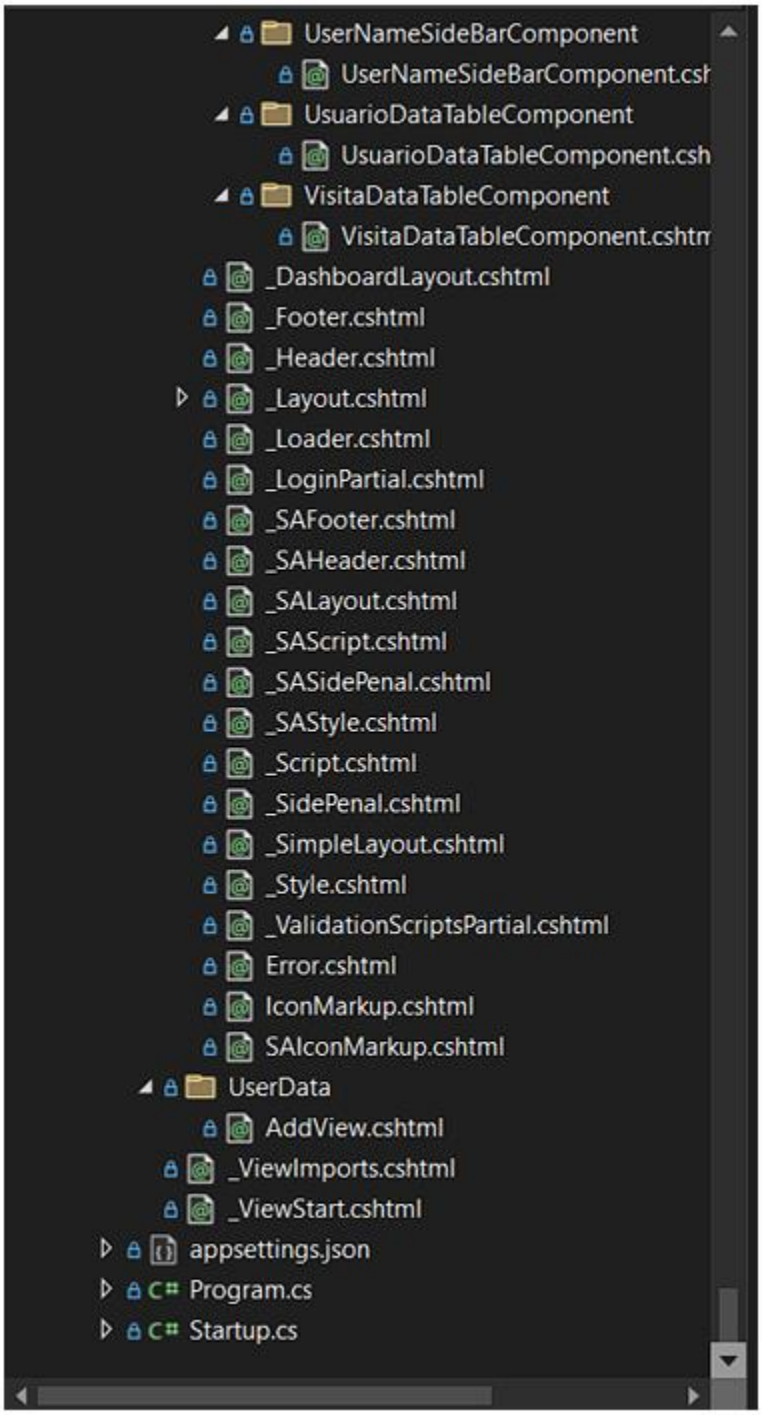

AddView.cshtml
EditView.cshtml
VisitaDataTableView.cshtml
Controllers
AccountController.cs
CiudadesController.cs
HomeController.cs
ProvinciasController.cs
UserDataController.cs
Models
AccountViewModels.cs
ErrorViewModel.cs
Services
Implementations
ApiServices.cs
PopulateSelectListServices.cs
Interfaces
IApiServices.cs
IPopulateSelectListServices.cs
ViewComponents
ActionAreaButtonsComponent.cs
ActionButtonsComponent.cs
AddAlicuotaConfiguracionComponent.cs
AddAlicuotaConfiguracionFromUnitCompo
AddAreaComponent.cs
AddCondominioComponent.cs
AddCondominioUsuariosComponent.cs
AddCondominioUsuariosDataTableCompor
AddDireccionComponent.cs
AddDirectivaAreaComponent.cs
AddDirectivaCondominioComponent.cs
AddGastoRegistroComponent.cs
AddGastosConfiguracionComponent.cs
AddIncidenciaComponent.cs


AddIncidenciaCondominioComponent.cs
AddNotaAdminComponent.cs
AddPeriodoDirectivaCondominioCompone
AddReservaComponent.cs
AddUnidadComponent.cs
AddUnidadUsuarioComponent.cs
AddUserProfileComponent.cs
AddUsuarioComponent.cs
AddUsuarioCondominioComponent.cs
AddVisitaComponent.cs
AdminSidePanelComponent.cs
AlicuotaConfiguracionDataTableComponer
AlicuotaRegistroDataTableComponent.cs
AlicuotasConfiguracionUnidadInfoCompon
AlicuotasRegistroUnidadInfoComponent.cs
AreaCardInfoComponent.cs
AreaChartComponent.cs
AreaInfoComponent.cs
AreaUsuarioTableComponent.cs
BarChartComponent.cs
CardCondominioComponent.cs
CondominioCardInformationComponent.cs
CondominioInfoComponent.cs
CondominiosEliminadosDataTableCompon
CondominioUsuariosDataTableComponent.
DashboardTotalsComponent.cs
DataTableToSelectUsuarioComponent.cs
DireccionInfoComponent.cs
DirectivaAreaDataTableComponent.cs
DirectivaCondominioDataTableComponent
EditAlicuotaConfiguracionComponent.cs
EditAreaComponent.cs
EditAreaInfoComponent.cs
EditCondominioComponent.cs
EditDireccionComponent.cs
EditGastosConfiguracionComponent.cs
EditGastosRegistrosComponent.cs
EditIncidenciaComponent.cs
EditIncidenciaCondominioComponent.cs
EditNotasAdminComponent.cs
EditPeriodoComponent.cs
EditReservaComponent.cs
EditUnidadComponent.cs
EditUserProfileComponent.cs
EditUsuarioComponent.cs
EditVisitaComponent.cs
GastosConfiguracionDataTableComponer
GastosRegistroDataTableComponent.cs
GastosRegistroDetailDataTableCompone
IncidenciaDataTableComponent.cs
IncidenciasCondominioDataTableCompor
InfoCondoComponent.cs
NotasAdminInfoComponent.cs
PeriodoDataTableComponent.cs
PeriodoDataTableToSelectDirectivaAreaC
PeriodoDataTableToSelectDirectivaComp
PeriodoInfoComponent.cs
PieChartComponent.cs
RadialChartComponent.cs
ReservaBarChartComponent.cs
ReservaDataTableComponent.cs
ReservaRadialChartComponent.cs
UnidadCardInfoComponent.cs
UnidadInfoComponent.cs
UnidadUsuariosInfoComponent.cs
UserNameSideBarComponent.cs
UsuarioDataTableComponent.cs
VisitaDataTableComponent.cs


Views
Account
ActivateAccount.cshtml
ActivateAccountFail.cshtml
ExpiredView.cshtml
ForgetPassword.cshtml
ForgotPasswordConfirmation.cshtml
Login.cshtml
PasswordResetSuccess.cshtml
PrivacyPolicy.cshtml
Register.cshtml
RegisterConfirmation.cshtml
ResetPassword.cshtml
Home
Contacto.cshtml
Index.cshtml
Planes.cshtml
Privacy.cshtml
Shared
Components
ActionAreaButtonsComponent
ActionAreaButtonsComponent.cs
ActionButtonsComponent
ActionButtonsComponent.cshtml
AddAlicuotaConfiguracionCompone
AddAlicuotaConfiguracionFromUnit
AddAlicuotaConfiguracionFromU
AddAreaComponent
AddAreaComponent.cshtml
AddAreaUsuarioComponent
AddAreaUsuarioComponent.csht
AddCondominioComponent
AddCondominioUsuariosComponen
AddCondominioUsuariosCompor
AddCondominioUsuariosDataTableC
AddCondominioUsuariosDataTab
AddDireccionComponent
AddDireccionComponent.cshtml
AddDirectivaAreaComponent
AddDirectivaAreaComponent.csh
AddDirectivaCondominioComponen
AddDirectivaCondominioCompo
AddGastoRegistroComponent
AddGastoRegistroComponent.csh
AddGastosConfiguracionComponent
AddGastosConfiguracionCompor
AddIncidenciaComponent
AddIncidenciaComponent.cshtml
AddIncidenciaCondominioCompone
AddIncidenciaCondominioComp
AddNotaAdminComponent
AddNotaAdminComponent.cshtn
AddPeriodoDirectivaCondominioCo
AddPeriodoDirectivaCondominio
AddReservaComponent
AddReservaComponent.cshtml
AddUnidadComponent
AddUnidadComponent.cshtml
AddUnidadUsuarioComponent
AddUnidadUsuarioComponent.cs
AddUserProfileComponent
AddUserProfileComponent.cshtm
AddUsuarioComponent
AddUsuarioComponent.cshtml
AddUsuarioCondominioComponent


AddVisitaComponent
AddVisitaComponent.cshtml
AdminSidePanelComponent
AdminSidePanelComponent.cshtr
AlicuotaConfiguracionDataTableCon
AlicuotaConfiguracionDataTable(
AlicuotaRegistroDataTableCompone
AlicuotaRegistroDataTableComp(
AlicuotasConfiguracionUnidadInfoC(
AlicuotasConfiguracionUnidadInf
AlicuotasRegistroUnidadInfoCompo
AlicuotasRegistroUnidadInfoCom
AreaCardInfoComponent
AreaCardInfoComponent.cshtml
AreaChartComponent
AreaChartComponent.cshtml
AreaInfoComponent
AreaInfoComponent.cshtml
AreaUsuarioTableComponent
AreaUsuarioTableComponent.csh
BarChartComponent
BarChartComponent.cshtml
CardCondominioComponent
CardCondominioComponent.csht
CondominioCardInformationCompo
CondominioCardInformationCom
CondominioInfoComponent
CondominioInfoComponent.cshtr
CondominiosEliminadosDataTableC(
CondominiosEliminadosDataTabl
CondominioUsuariosDataTableComp
CondominioUsuariosDataTableC(
DashboardTotalsComponent
DashboardTotalsComponent.csht
DireccionInfoComponent
DireccionInfoComponent.cshtml
DirectivaAreaDataTableComponent
DirectivaAreaDataTableCompone
DirectivaCondominioDataTableCom
DirectivaCondominioDataTableC(
EditAlicuotaConfiguracionCompone
EditAlicuotaConfiguracionCompo
EditAreaComponent
EditAreaComponent.cshtml
EditAreaInfoComponent
EditAreaInfoComponent.cshtml
EditCondominioComponent
EditCondominioComponent.cshtr
EditDireccionComponent
EditDireccionComponent.cshtml
EditGastosConfiguracionComponent
EditGastosConfiguracionCompon
EditGastosRegistrosComponent
EditGastosRegistrosComponent.c
EditIncidenciaComponent
EditIncidenciaComponent.cshtml
EditIncidenciaCondominioCompone
EditIncidenciaCondominioCompo
EditNotasAdminComponent
EditNotasAdminComponent.cshtr
EditPeriodoComponent
EditPeriodoComponent.cshtml
EditReservaComponent
EditReservaComponent.cshtml
EditUnidadComponent
EditUnidadComponent.cshtml
EditUserProfileComponent
EditUserProfileComponent.cshtm


EditUserProfileComponent.cshtm
EditUsuarioComponent
EditUsuarioComponent.cshtml
EditVisitaComponent
EditVisitaComponent.cshtml
GastosConfiguracionDataTableComp
GastosConfiguracionDataTableCc
GastosRegistroDataTableComponen
GastosRegistroDataTableCompor
GastosRegistroDetailDataTableComp
GastosRegistroDetailDataTableCc
IncidenciaDataTableComponent
IncidenciaDataTableComponent.c
IncidenciasCondominioDataTableCo
IncidenciasCondominioDataTable
InfoCondoComponent
InfoCondoComponent.cshtml
NotasAdminInfoComponent
NotasAdminInfoComponent.cshtı
PeriodoDataTableComponent
PeriodoDataTableComponent.csh
PeriodoDataTableToSelectDirectivaA
PeriodoDataTableToSelectDirecti
PeriodoDataTableToSelectDirectivaC
PeriodoDataTableToSelectDirecti
PeriodoInfoComponent
PeriodoInfoComponent.cshtml
PieChartComponent
PieChartComponent.cshtml
RadialChartComponent
RadialChartComponent.cshtml
ReservaBarChartComponent
ReservaBarChartComponent.cshtr
ReservaDataTableComponent
ReservaDataTableComponent.csh
ReservaRadialChartComponent
ReservaRadialChartComponent.cs
UnidadCardInfoComponent
UnidadCardInfoComponent.cshtn
UnidadInfoComponent
UnidadInfoComponent.cshtml
UnidadUsuariosInfoComponent
UnidadUsuariosInfoComponent.c
UserNameSideBarComponent
UserNameSideBarComponent.csh
UsuarioDataTableComponent
UsuarioDataTableComponent.csh
VisitaDataTableComponent
VisitaDataTableComponent.cshtm
_DashboardLayout.cshtml
_Footer.cshtml
_Header.cshtml
_Layout.cshtml
_Loader.cshtml
_LoginPartial.cshtml
_SAFooter.cshtml
_SAHeader.cshtml
_SALayout.cshtml
_SAScript.cshtml
_SASidePenal.cshtml
_SAStyle.cshtml
_Script.cshtml
_SidePenal.cshtml
_SimpleLayout.cshtml
_Style.cshtml
_ValidationScriptsPartial.cshtml
Error.cshtml


UserNameSideBarComponent
UserNameSideBarComponent.csh
UsuarioDataTableComponent
UsuarioDataTableComponent.csh
VisitaDataTableComponent
VisitaDataTableComponent.cshtm
_DashboardLayout.cshtml
_Footer.cshtml
_Header.cshtml
_Layout.cshtml
_Loader.cshtml
_LoginPartial.cshtml
_SAFooter.cshtml
_SAHeader.cshtml
_SALayout.cshtml
_SAScript.cshtml
_SASidePenal.cshtml
_SAStyle.cshtml
_Script.cshtml
_SidePenal.cshtml
_SimpleLayout.cshtml
_Style.cshtml
_ValidationScriptsPartial.cshtml
Error.cshtml
IconMarkup.cshtml
SAIconMarkup.cshtml
UserData
AddView.cshtml
_ViewImports.cshtml
_ViewStart.cshtml
appsettings.json
Program.cs
Startup.cs

A continuación, se muestra la evidencia del código respectivo a la página principal de la aplicación:

```
@model List<Services.DTOs.Response.CondominioResponseDTO>
<style>
    .area-card {
        box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.2), 0 6px 10px 0 rgba(0, 0, 0, 0.10);
        /* Estilo de sombra */
        padding-top: 20px;
        padding-bottom: 20px;
        border-radius: 10px;
        transition: transform 0.3s ease;
    }

        .area-card:hover {
            transform: scale(1.05); /* Hace zoom en un 5% al pasar el mouse sobre la tarjeta */
        }
</style>
<div class="container-fluid">
    <div class="row">
        @foreach (var condominio in Model)
        {
            <div class="col-xl-4 col-md-6 mb-4">
                @{
                    string imageSrc = "";
                    if (condominio.ImageData != null && condominio.ImageData.Length > 0)
                    {
                        string base64Image = Convert.ToBase64String(condominio.ImageData);
                        imageSrc = $"{base64Image}";
                    }
                    else
                    {
                        imageSrc = "/assets/images/plaza10.png";
                    }
                }
                <a class="loadContentView" href="#"
                   data-url-action="@(Url.Action("Index", "Condominio", new { area = "Admin", Id = condominio.Id }))"
                   data-content-id="#maincontent"
                   data-params='{"DireccionId":"@(condominio.DireccionId ?? -1)"}'
                   data-error-title="@Common.Resources.FileResource.TEXT_ERROR"
                   data-error-text="@Common.Resources.FileResource.VAL_LOADING_FORM_FAIL"
                   data-fail_redirect_url="@Url.Action("Index", "Dashboard", new { area = "Admin" })">
                    <div class="card custom-card area-card">
                        <div class="card-header pb-0" style="overflow: hidden;">
```

Las siguientes imágenes corresponden a la vista del formulario para crear un condominio dentro del sistema:

```
@model Services.DTOs.Request.CondominioRequestDTO


<div class="page-body">
    <!-- Container-fluid starts-->
    <div class="container-fluid">
        <div class="edit-profile">
            <div class="row">
                <div class="col-xl-12">
                    <form id="condominioForm" asp-controller="CondominioData" asp-action="AddUpdateCondominio" asp-area="" method="post" class="card needs-validation" nova
                        @Html.AntiForgeryToken()
                        <div id="validation-summary" class="alert alert-danger d-none" role="alert">
                            <ul></ul>
                        </div>

                        <div class="card-header pb-0">
                            <h4 class="card-title mb-0">@Common.Resources.FileResource.TEXT_CREATE_CONDOMINIO</h4>
                        </div>
                        <div class="card-body">
                            <div class="row">

                                <div class="col-sm-12 col-md-4">
                                    <div class="mb-3">
                                        <label class="form-label">@Common.Resources.FileResource.TEXT_NAME</label>
                                        @Html.TextBoxFor(m => m.Nombre, new { id = "condoTextBox", autocomplete = "off", @class = "form-control", maxlength = 100, required
                                    </div>
                                </div>
                                <div class="col-sm-12 col-md-8">
                                    <div class="mb-3">
                                        <label class="form-label">@Common.Resources.FileResource.TEXT_DESCRIPTION</label>
                                        @Html.TextBoxFor(m => m.Descripcion, new { id = "descriptionTextBox", autocomplete = "off", @class = "form-control", maxlength = 30
                                    </div>
                                </div>
                            </div>

                        </div>
                        <div class="card-footer text-end">
                            <button class="btn btn-sm btn-light loadContentView pull-left"
                                    data-url-action="@Url.Action("Dashboard", "Dashboard", new { area = "Admin" })"
                                    data-content-id="#maincontent"
```

```
                    </div>
                    <div class="card-footer text-end">
                        <button class="btn btn-sm btn-light loadContentView pull-left"
                                data-url-action="@Url.Action("Dashboard", "Dashboard", new { area = "Admin" })"
                                data-content-id="#maincontent"
                                data-params=""
                                data-error-title="@Common.Resources.FileResource.TEXT_ERROR"
                                data-error-text="@Common.Resources.FileResource.VAL_LOADING_FORM_FAIL">
                            @Common.Resources.FileResource.TEXT_CANCEL
                        </button>

                        <button id="submitButton" class="btn btn-sm btn-primary submitButton"
                                data-url-action="@Url.Action("AddCondominio", "Condominio", new { area = "Admin" })"
                                data-form-id="#condominioForm"
                                data-sweet-title="@Common.Resources.FileResource.VAL_DATA_SAVED"
                                data-sweet-message="@Common.Resources.FileResource.VAL_DATA_SAVED_DESCRIPTION"
                                data-success-redirect-url='@Url.Action("Dashboard", "Dashboard", new { area = "Admin" })'
                                data-success-content-id="#maincontent"
                                data-sucess-params=""
                                data-error-title="@Common.Resources.FileResource.TEXT_ERROR"
                                data-error-text="@Common.Resources.FileResource.VAL_LOADING_FORM_FAIL">
                            @Common.Resources.FileResource.TEXT_SAVE
                        </button>
                    </div>
                </form>
            </div>
        </div>
    </div>
    <!-- Container-fluid Ends-->
</div>

<!-- Plugins JS datepicker start-->
<script src="../assets/js/datepicker/date-picker/datepicker.js"></script>
<script src="../assets/js/datepicker/date-picker/datepicker.en.js"></script>
<script src="../assets/js/datepicker/date-picker/datepicker.custom.js"></script>
<script src="../assets/js/tooltip-init.js"></script>
```

Como se plasman en las siguientes imágenes este es el código respectivo a la creación de las tablas con la información de las reservas ingresadas dentro del sistema:

```
@model List<Services.DTOs.Response.ReservaResponseDTO>

<div class="card">
        <div class="card-header">
            <h5>@Common.Resources.FileResource.TEXT_RESERVAS_AREAS</h5>
            <button class="btn btn-primary btn-xs loadContentView pull-right" title="@Common.Resources.FileResource.TEXT_ADD_RESIDENTE" type="button"
                    data-url-action="@(Url.Action("AddView", "Reserva", new { area = "Admin", condominioId = ViewBag.CONDOMINIOID }))"
                    data-content-id="#actions-section"
                    data-params=""
                    data-success-redirect-url='@Url.Action("ReservaDataTableView", "Reserva", new { area = "" })'
                    data-error-title="@Common.Resources.FileResource.TEXT_ERROR"
                    data-error-text="@Common.Resources.FileResource.VAL_LOADING_FORM_FAIL"
                    data-fail_redirect_url="@Url.Action("ReservaDataTableView", "Reserva", new { area = "" })">
                @Common.Resources.FileResource.TEXT_ADD_RESERVA
            </button>
        </div>
    <div class="card-body">
        <div class="table-responsive">
            <table class="display" id="reservasDatatable">
                <thead>
                    <tr>
                        <th>@Common.Resources.FileResource.TEXT_APROBADO</th>
                        <th>@Common.Resources.FileResource.TEXT_AREA</th>
                        <th>@Common.Resources.FileResource.TEXT_USUARIO_RESERVA</th>
                        <th>@Common.Resources.FileResource.TEXT_IDENTIFICATION</th>
                        <th>@Common.Resources.FileResource.TEXT_FECHA_INICIO_RESERVA</th>
                        <th>@Common.Resources.FileResource.TEXT_HORA_INICIO_RESERVA</th>
                        <th>@Common.Resources.FileResource.TEXT_HORA_FIN_RESERVA</th>
                        <th>@Common.Resources.FileResource.TEXT_ACTIONS</th>
                    </tr>
                </thead>
                <tbody>
                    @foreach (var reserva in Model)
                    {
                        <tr>
                            <td>
                                <input id="chk-ani-@reserva.Aprobado" @(reserva.Aprobado == true ? "checked" : "") style="height:25px"
                                data-url = "@Url.Action("Edit","Reserva",new {area = "Admin"})"
                                data-reserva-id="@reserva.Id" class="checkbox_animated checkbox-reserva" type="checkbox" />
                            </td>
                            <td>@reserva.AreaNombre</td>
```

```
                        <td>@(reserva.Usuario.Identificacion)</td>
                        <td>@reserva.FechaInicio</td>
                        <td>@reserva.HoraInicio</td>
                        <td>@reserva.HoraFin</td>
                        <td>
                            <a class="text-primary loadContentView" href="#" title="@Common.Resources.FileResource.TEXT_EDIT"
                               data-url-action="@(Url.Action("EditView", "Reserva", new { area = "Admin", Id = reserva.Id }))"
                               data-content-id="#actions-section"
                               data-params=""
                               data-error-title="@Common.Resources.FileResource.TEXT_ERROR"
                               data-error-text="@Common.Resources.FileResource.VAL_LOADING_FORM_FAIL"
                               data-fail_redirect_url="@Url.Action("DataTableView", "Reserva", new { area = "" })"><i class="fa fa-edit m-r-5"></i></a>

                            <a class="text-danger m-l-5 deleteButton" href="#" title="@Common.Resources.FileResource.TEXT_DELETE"
                               data-url-action="@Url.Action("Delete", "Reserva", new { area = "Admin", Id = reserva.Id })"
                               data-delete-confirm="@Common.Resources.FileResource.TEXT_DELETE_CONFIRMATIONS"
                               data-delete-warning="@Common.Resources.FileResource.TEXT_DELETE_WARNING"
                               data-delete-success="@Common.Resources.FileResource.TEXT_DELETE_SUCCESS"
                               data-success-redirect-url="@Url.Action("ReservaDataTableView", "Reserva", new { area = "Admin", condominioId = ViewBag.CONDOMINIOID })"
                               data-success-content-id="#actions-section"
                               data-error-title="@Common.Resources.FileResource.TEXT_ERROR"
                               data-error-text="@Common.Resources.FileResource.ERR_DELETE_FAIL"
                               data-delete-cancel-text="@Common.Resources.FileResource.TEXT_DELETE_ACTION_CANCEL"><i class="fa fa-trash-o m-r-5"></i></a>

                        </td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
</div>
<div class="card-footer">
    <button class="btn btn-xs btn-light clearSectionContentView pull-left"
            data-clear-content-id="#actions-section">
        @Common.Resources.FileResource.TEXT_CANCEL
    </button>
</div>
</div>

<script>
    $('#reservasDatatable').DataTable();
</script>
```

A través de las siguientes se evidencia el código para el registro de la información de los usuarios:

```
@model List<Services.DTOs.Response.UsuarioResponseDTO>
@{
    ViewData["Title"] = @Common.Resources.FileResource.TEXT_INFO_USUARIOS;
}

<div class="page-body">

    <div class="col-sm-12">
        <div class="card">
            <div class="card-header">
                <h5>@Common.Resources.FileResource.TEXT_INFO_USUARIO</h5>
                <button class="btn btn-primary btn-xs m-b-5 loadContentView pull-right" title="@Common.Resources.FileResource.TEXT_ADD_RESIDENTE" type="button"
                        data-url-action="@(Url.Action("AddView", "Usuario", new { area = "Admin" }))"
                        data-content-id="#maincontent"
                        data-params=""
                        data-success-redirect-url='@Url.Action("DataTableView", "UserData", new { area = "" })'
                        data-error-title="@Common.Resources.FileResource.TEXT_ERROR"
                        data-error-text="@Common.Resources.FileResource.VAL_LOADING_FORM_FAIL"
                        data-fail_redirect_url="@Url.Action("DataTableView", "UserData", new { area = "" })">
                    @Common.Resources.FileResource.TEXT_ADD_USUARIO
                </button>
            </div>
            <div class="card-body">
                <div class="table-responsive">

                    <table class="display" id="basic-1">
                        <thead>
                            <tr>
                                <th>@Common.Resources.FileResource.TEXT_IDENTIFICATION</th>
                                <th>@Common.Resources.FileResource.TEXT_NAME</th>
                                <th>@Common.Resources.FileResource.TEXT_LASTNAME</th>
                                <th>@Common.Resources.FileResource.TEXT_GENDER</th>
                                <th>@Common.Resources.FileResource.TEXT_BORNDATE</th>
                                <th>@Common.Resources.FileResource.TEXT_NATIONALITY</th>
                                <th>@Common.Resources.FileResource.TEXT_TELEFONO</th>
                                <th>@Common.Resources.FileResource.TEXT_EMAIL</th>
                                <th>@Common.Resources.FileResource.TEXT_UNIDAD_TYPE</th>
                                <th>@Common.Resources.FileResource.TEXT_ACTIONS</th>
                            </tr>
                        </thead>
                        <tbody>
```

```
                    </thead>
                    <tbody>
                        @foreach (var usuario in Model)
                        {
                            <tr>
                                <td>@usuario.Identificacion</td>
                                <td>@usuario.Nombre</td>
                                <td>@usuario.Apellidos</td>
                                <td>@(usuario.Genero == "M" ? "Masculino" : (usuario.Genero == "F" ? "Femenino" : "Otro"))</td>
                                <td>@usuario.FechaNacimiento.ToShortDateString()</td>
                                <td>@usuario.NacionalidadNombre</td>
                                <td>@usuario.Telefono</td>
                                <td>@usuario.Email</td>
                                <td>@(usuario.TipoRelacionUnidadNombre)</td>

                                <td>
                                    <a class="text-primary loadContentView" href="#" title="@Common.Resources.FileResource.TEXT_EDIT"
                                       data-url-action="@(Url.Action("EditView", "Usuario", new { area = "Admin", Id = usuario.Id }))"
                                       data-content-id="#maincontent"
                                       data-params=""
                                       data-error-title="@Common.Resources.FileResource.TEXT_ERROR"
                                       data-error-text="@Common.Resources.FileResource.VAL_LOADING_FORM_FAIL"
                                       data-fail_redirect_url="@Url.Action("DataTableView", "UserData", new { area = "" })"><i class="fa fa-edit m-r-5"></i></a>

                                    <a class="text-danger m-l-5 deleteButton" href="#" title="@Common.Resources.FileResource.TEXT_DELETE"
                                       data-url-action="@Url.Action("Delete", "UserData", new { area = "", Id = usuario.Id })"
                                       data-delete-confirm="@Common.Resources.FileResource.TEXT_DELETE_CONFIRMATIONS"
                                       data-delete-warning="@Common.Resources.FileResource.TEXT_DELETE_WARNING"
                                       data-delete-success="@Common.Resources.FileResource.TEXT_DELETE_SUCCESS"
                                       data-success-redirect-url="@Url.Action("DataTableView", "UserData", new { area = "" })"
                                       data-success-content-id="#maincontent"
                                       data-error-title="@Common.Resources.FileResource.TEXT_ERROR"
                                       data-error-text="@Common.Resources.FileResource.ERR_DELETE_FAIL"
                                       data-delete-cancel-text="@Common.Resources.FileResource.TEXT_DELETE_ACTION_CANCEL"><i class="fa fa-trash-o m-r-5"></i></a>

                                </td>
                            </tr>
                        }
                    </tbody>
                </table>
            </div>
```

8. Referencias

Alfaro, I. (2018). *Nuevos condominios, nuevas administraciones.* Ariel.

Arrioja, N. (2018). *Curso de Programacion C#.* Users.

Barbettini, N. (2018). *El pequeño libro de ASP.NET Core.* Recuperado el 7 de abril de 2024, de aspnetcoremaster.com: https://aspnetcoremaster.com/little-aspnetcore-book/ElPequenoLibroDeASPNETCore.pdf

Bascón, E. (2017). El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing. *Acta Nova, 2*(4), 1-12. Recuperado el 19 de agosto de 2024, de http://www.scielo.org.bo/scielo.php?script=sci_arttext&pid=S1683-07892004000100005

Buri, C. (2021). *"Manejo óptimo de la gestión económica y administrativa de los condominios que se encuentran bajo el régimen de propiedad horizontal en la Ciudad De Guayaquil.* Universidad Católica de Santiago de Guayaquil .

Celí, J., Boné, F., & Mora, P. (2023). *Programación Web. Del Frontend al Backend.* Grupo AEA.

Choto, K. (2018). *Modelo de gestión para la administración de condominios habitacionales en el Distrito Metropolitano de Quito.* UCE.

Congreso Nacional . (2005). *Ley de Propiedad Horizontal.* Registro oficial Codificación 2005-013.

Contribuidores de Wikipedia. (17 de 07 de 2015). *Investigación*. Obtenido de Wikipedia, La enciclopedia libre: https://es.wikipedia.org/wiki/Investigaci%C3%B3n

Coronado, C. (s.f.).

Coronado, C. (2019). *Desarrollo de un sistema web; fortalecimiento de los procesos de gestión administrativa y financiera; condominio Solar del Río; Ciudad De Ibarra; Microsoft Azure.* UTA.

De Santiago, J. (2023). *Desarrollo de API REST y Frontend dedicada al mundo de los acuarios.* Universidad Complutense de Madrid.

Escofet, C. (2019). El lenguaje SQL. FUOC

Espinosa, C. (2018). Propuesta de solución tecnológica para la medición real de la satisfacción del cliente mediante reconocimiento facial. *Revista Ingeniantes, 3*(2), 89-94. Recuperado el 20 de agosto de 2024, de https://citt.itsm.edu.mx/ingeniantes/articulos/ingeniantes5no2vol1/15%20Propuesta%20de%20solucion%20tecnologica%20para%20la%20medicion.pdf

Gamarra, F. (2023). *Visual Studio Code.* Users.

García, M. (2019). Métodos de muestreo en investigación científica. *Métodos de Investigación, 7*(2), 45-59.

Gavilánez, Ó., & Layedra, N. (2022). nálisis comparativo de Patrones de Diseño de Software. *Polo del Conocimiento, 7*(7), 2145-2565. doi:10.23857/pc.v7i7

Gracia del Busto, H., y Yanes, O. (2012). Bases de datos NoSQL. *Revista Telematica, 11*(3), 21-33. Retrieved 23 de agosto de 2024, from http://revistatelematica.cujae.edu.cu/index.php/teleBases de datos NoSQLIng

Gerginov, R. (2020). *El gran libro de HTML5, CSS3 y JavaScript.* Marcombo.

Giglia, Á. (2016). La democracia en la vida cotidiana.Dos casos de gestión de condominios en la ciudad de México. *Alteridades, 6*(11), 75-85. Recuperado el 1 de marzo de 2024, de https://www.redalyc.org/pdf/747/74711339007.pdf

Gómez, J. (2023). *El desarrollo web desde el entorno del cliente.* ALPHAEDITORIAL ESIC.

González, E. (2023). *Análisis comparativo de patrones de diseño MVC y MVP para el rendimiento de aplicaciones web.* Universidad de Lima.

Gracida, C. M. (2022). *El condominio, su operación y administración.* IMCP.

Hernández, J. (2017). *API.* Univesidad del Suroeste de México.

Hernández, M., & Baquero, L. E. (2021). *Fundamentos de Programación Web.* Universidad ECCI .

Hernández, R. (2001). *Metodología de Investigación.* México: McGraw_Hill.

Hernández, R. (28 de junio de 2021). *El patrón modelo-vista-controlador: Arquitectura y frameworks explicados*. Recuperado el 19 de agosto de 2024, de www.freecodecamp.org: https://www.freecodecamp.org/espanol/news/el-modelo-de-arquitectura-view-controller-pattern/

Hernández, R., Fernández, C., & Baptista, M. d. (2017). *Metodología de la Investigación.* México D.F: McGraw-Hill / Interamericana Editores.

Hidalgo, R. (2019). De los pequeños condominios a la ciudad vallada: las urbanizaciones cerradas y la nueva geografía social en Santiago de Chile. *EURE, 30*(21), 29-52. doi:http://dx.doi.org/10.4067/S0250-71612004009100003

Hinojosa, P., & Merelo, J. (2017). *Aprende Git: ... y, de camino, GitHub* . Mundo.

Martínez, I. (2021). *Hosting y dominios.* Recuperado el 17 de marzo de 2024, de www.dit.upm.es: https://www.dit.upm.es/~imartinez/CursoWebETSAM/files/HostingDominio.pdf

Mattos, J. (2021). *Aplicación web para obtener la AFP que brinda la mayor rentabilidad mediante WinAutomation Console, HTML, CSS y Win SCP* . Universidad Nacional Mayor de San Marcos .

Méndez, C. E. (2002). *Metodología de Investigación, guía para la elaboración de Proyectos.* México: McGraw-Hill.

Meyer, K., & Jurgen, B. (2020). a difusión de condominios en las metrópolis Latinoamericanas. El ejemplo de Santiago de Chile. *Revista De Geografía Norte Grande,, I*(32), 39-53. Recuperado el 7 de marzo de 2024, de https://revistacienciapolitica.uc.cl/index.php/RGNG/article/view/43579

Molina, J. (2021). *Metodologías ágiles de desarrollo aplicadas a la enseñanza de la programación.* Universidad de Alicante.

Montalvo, A. P., & Paredes, D. A. (2023). *Desarrollo de un sistema software multiplataforma basado en un modelo de gestión administrativa enmarcado en la arquitectura REST y REDUX para la optimización de la administración del conjunto habitacional "Oriental"*. ESPE.

Montoya, I. (2016). *Condominio modelo de organización administrativa.* Recuperado el 7 de marzo de 2024, de Universidad Nacional de Colombia: file:///C:/Users/alcid/Downloads/Condominios.ModelodeorganizacionMontoyaAMontoyaI2016.pdf

Mora, A. (2014). *Bases de datos. Diseño y gestión.* Síntesis.

Moreno, J. C. (2018). *Entornos de desarrollo integrado.* Síntesis.

Moure, B. (2023). *Git y GitHub desde cero: Guía de estudio teórico-práctica paso a paso más curso en vídeo.* Mundo.

Muñoz, M. (2019). *Entity Framework Core: Manual de estudiante.* MVP.

Nolasco, J. S. (2018). *Desarrollo de aplicaciones móviles con Android 2ª* . Ediciones de la U.

Ockert, D. P. (2022). *Visual Studio 2022 en profundidad: explore las fantásticas funciones de Visual Studio 2022.* Users.

Pantaleo, G., & Rinaudo, L. (2017). *Ingenieria De SoftwareIngenieria De Software.* Alfaomega.

Pavón, J. (2023). *Patrones de diseño orientado a objetos.* Universidad Complutense Madrid.

Paz, A. (2017). *Control de Versiones de Software con GIT.* IT Campus Academy.

Puerta, J. M. (2018). *Desarrollo de una API para la descripción y gestión de Servicios Web REST* . Universidad Jaume.

Real Academia de la Lengua Española. (2018). *Dicionario de la Lengua Española.* ESPASA.

Revilla, E. (2019). *Desarrollo de aplicaciones móviles multiplataforma y PWAs con Ionic y Firebase desde cero: Aprende a crear apps para Android, IOS y PWAs, de manera sencilla.*

Rubiales, M. (2021). *Curso de desarrollo web: HTML, CSS Y JAVASCRIPT.* Anaya multimedia.

Salas, R. (2022). *Dise˜ no y desarrollo de API y servicios web RESTful para la gesti´ on de Chatbot multiplataforma: la UEx como caso de estudio.* Universidad de Extremadura.

Sepúlveda, L. (2023). *Control de versiones para la gestión de archivos digitales.* EAE.

Serrano, Á. (2017). *Condominio.* UNAM.

Serrano, Á. (2017,p.3). *Condominio.* UNAM.

Sestoft, P. (2022). *Lenguajes de programación conceptos. Segunda edición .* UTICS.

Silberschatz, A., Korth, H., & Sudarshan, S. (2019). *Fundamentos de bases de datos. Cuarta edición .* McGRAW-HILL.

Synology. (s.f.). *¿Qué es NAS?* Recuperado el 12 de Noviembre de 2023, de Synology: https://www.synology.com/es-mx/dsm/solution/what-is-nas/for-home

Torres, M. (2017). *Desarrollo de aplicaciones móviles con Android.* Marcombo.

Unir. (22 de septiembre de 2022). *Framework: qué es, para qué sirve y algunos ejemplos.* Recuperado el 11 de abril de 2024, de unirfp.unir.net: https://unirfp.unir.net/revista/ingenieria-y-tecnologia/framework/

Vázquez, G. (2017). *Régimen de ñpropiedad en condominio.* Universidad Autonóma de Puebla.

Vicente, R., & Vivas, J. (2014). *Introducción a las Bases de Datos Un enfoque basado en casos de estudio.* GIDSAW.

Vizcaíno, A., García, F., & Piattini, M. (2021). *Desarrollo global de Software.* Ra-ma Editorial.

ANEXOS

Anexo 1: Cuestionario que conforma la encuesta aplicada de manera directa a una muestra de 163 personas integrada por los residentes, guardias de seguridad y directiva del edificio Plaza 10 para conocer necesidades a tener en cuenta en un sistema de condominio denominado SCC.

Anexo 2: Muestras de encuestas

Anexo 3:Documentación oficial de ASP.NET Core

Anexo 4: Documentación oficial de Visual Studio 2022

Anexo 5: Documentación oficial de ASP.NET MVC