Proyecto de Grado

Presentado ante la ilustre Universidad de Los Andes como requisito final
para obtener el Título de Ingeniero de Sistemas

Desarrollo de una aplicación móvil que permita automatizar la gestión administrativa de condominios.

por

Br.Luzalba Sorelyz Rincón González
Tutor: Yaneth Moreno, M.Sc

Noviembre de 2022

Desarrollo de una aplicación móvil que permita automatizar la gestión administrativa de condominios.

Br. Luzalba Sorelyz Rincón González.
Escuela de Ingeniería de Sistemas, Universidad de Los Andes, 2022.

Resumen: La situación actual causada por el virus COVID-19 ha aumentado el uso de las aplicaciones móviles, a traves de ellas las personas pueden consultar información o realizar otras actividades manteniendo el distancionamiento social. En la administración de un condominio se necesitan tomar desiciones en conjunto con todos los propietarios, mantener informado sobre las situaciones que se presenten cotidianamente, entre otras. Por lo mencionado anteriormente, se genera la necesidad de desarrollar una aplicación móvil que permita automatizar la gestion administrativa de condominios ayudando a centralizar todos los procesos como consulta información, mensajeria, reservas areas comunes, consultar pagos entre otros. Para la ejecución del trabajo se utilizára la Metodología Mobile-D que ayudara a enfocarse más en la creación de la aplicación, tambien se usará el framework Ionic para el desarrollo del producto.

Palabras clave: Metodología Mobile-D , Aplicación móvil, Administración de Condominios, Desarrollo de Software.

Índice general

Índice de figuras **5**

Índice de cuadros **8**

1. Introducción **13**

1.1. **Antecedentes** . . . 14
1.2. **Planteamiento del Problema** . . . 15
1.3. **Justificación del Proyecto** . . . 16
1.4. **Objetivos del Proyecto** . . . 16
1.5. **Metodología** . . . 17
1.6. **Alcances** . . . 18

2. Marco Teórico **19**

2.1. **Condominios y Administración** . . . 19
2.1.1. **Aplicaciones para la administración de condominios** . . . 22
2.1.2. **Prinicipales características de la aplicación propuesta** . . . 24
2.1.3. **Características Diferenciadoras de la Aplicación Propuesta con las Aplicaciones Revisadas** . . . 25
2.2. **Desarrollo de Aplicaciones Móviles:** . . . 25
2.2.1. **Ecosistema móvil** . . . 27
2.2.2. **Fragmentación** . . . 28
2.3. **Herramientas y tecnología de desarrollo:** . . . 30
2.3.1. **Ionic** . . . 30
2.3.2. **Node** . . . 31
2.3.3. **Heroku** . . . 33
2.3.4. **Emuladores** . . . 33

2.3.5. **Postman** . . . 34
2.3.6. **One signal** . . . 35
2.3.7. **Base de Datos** . . . 36

3. Análisis y diseño de la Aplicación 39
3.1. Fase 1: Exploración . . . 39
3.1.1. Actividad 1: Obtención de los requisitos . . . 41
3.1.2. Actividad 2: Clasificación de los requisitos: . . . 44
3.2. Fase 2: Inicialización . . . 47
3.2.1. Actividad 2: Estructuración del Software . . . 47
3.2.2. Casos de uso, diagramas de secuencias y diagramas de actividades del sistema: . . . 49
3.2.3. Diseño de base de datos . . . 88

4. Implementación y Pruebas 98
4.1. Fase 3: Producción . . . 98
4.1.1. Actividad 1: Codificación de la Aplicación . . . 98
4.1.2. Actividad 2: Codificación del API REST . . . 101
4.2. Fase 5: Pruebas . . . 102
4.2.1. Actividad 1: Emulación y simulación: . . . 104
4.2.2. Actividad 2: Dispositivos Reales: . . . 105
4.2.3. Actividad 3: Pruebas Funcionales . . . 106

5. Conclusiones y Recomendaciones 113
5.1. Conclusiones . . . 113
5.2. Recomendaciones . . . 113

Referencias 115

Referencias 115

Índice de figuras

1.1. Metodología MOBIL-D 18

3.1. Diagrama de Actores. 48
3.2. Diagrama de Casos de Uso 01 - Iniciar sesión, crear cuenta y recuperar contraseña. 50
3.3. Diagrama de Actividades Caso de uso CU-01. Crear cuenta 53
3.4. Diagrama de Actividades Caso de uso CU-02. Iniciar sesión 54
3.6. Diagrama de Secuencia Caso de uso CU-01. Crear cuenta 54
3.5. Diagrama de Actividades Caso de uso CU-03. Recuperar contraseña . 55
3.7. Diagrama de Secuencia Caso de uso CU-02. Iniciar sesión 55
3.8. Diagrama de Secuencia Caso de uso CU-03. Recuperar contraseña . . 56
3.9. Diagrama de Caso de uso CU-03. Administrar perfil 57
3.10. Diagrama de Actividades Caso de uso CU-04. Editar perfil 60
3.11. Diagrama de Actividades Caso de uso CU-05. Cambiar contraseña . . 60
3.12. Diagrama de Actividades Caso de uso CU-06. Cerrar sesión 61
3.13. Diagrama de Secuencia Caso de uso CU-04. Editar perfil 61
3.14. Diagrama de Secuencia Caso de uso CU-05. Cambiar contraseña . . . 62
3.15. Diagrama de Caso de uso CU-03 Gestión de condominios 62
3.16. Diagrama de Actividad Caso de uso CU-07. Crear Condominio 66
3.17. Diagrama de Actividad Caso de uso CU-08. Buscar Condominio . . . 66
3.18. Diagrama de Actividad Caso de uso CU-09. Ver Condominio 67
3.19. Diagrama de Actividad Caso de uso CU-10. Editar Condominio . . . 67
3.20. Diagrama de Actividad Caso de uso CU-11. Eliminar Condominio . . 67
3.21. Diagrama de Secuencia Caso de uso CU-07. Crear Condominio 68
3.23. Diagrama de Secuencia Caso de uso CU-09. Ver Condominio 68

3.22. Diagrama de Secuencia Caso de uso CU-08. Buscar Condominio . . . 69
3.24. Diagrama de Secuencia Caso de uso CU-10. Editar Condominio . . . 69
3.25. Diagrama de Caso de uso CU-11. Eliminar Condominio 70
3.35. Diagrama de Caso de uso CU-05 Gestión de pagos 73
3.26. Diagrama de Caso de uso CU-12 Gestión de ingresos/egresos 74
3.27. Diagrama de Actividad Caso de uso CU-12. Crear Ingreso/Egreso . . 74
3.28. Diagrama de Actividad Caso de uso CU-13. Listar Ingreso/Egreso . . 75
3.29. Diagrama de Actividad Caso de uso CU-14. Editar Ingreso/Egreso . . 75
3.30. Diagrama de Actividad Caso de uso CU-15. Eliminar Ingreso/Egreso 78
3.31. Diagrama de Secuencia Caso de uso CU-12. Crear Ingreso/Egreso . . 78
3.36. Diagrama de Actividad Caso de uso CU-16. Listar Pago 78
3.32. Diagrama de Secuencia Caso de uso CU-13. Listar Ingreso/Egreso . . 79
3.33. Diagrama de Secuencia Caso de uso CU-14. Editar Ingreso/Egreso . . 79
3.37. Diagrama de Actividad Caso de uso CU-17. Ver Pago 79
3.34. Diagrama de Secuencia Caso de uso CU-15. Eliminar Ingreso/Egreso 80
3.38. Diagrama de Actividad Caso de uso CU-18. Enivar comprobante de Pago 80
3.39. Diagrama de Actividad Caso de uso CU-19. Aprobar Pago 81
3.40. Diagrama de Secuencia Caso de uso CU-16. Listar Pago 81
3.41. Diagrama de Secuencia Caso de uso CU-17. Ver Pago 82
3.42. Diagrama de Secuencia Caso de uso CU-18. Enivar comprobante de Pago 82
3.43. Diagrama de Secuencia Caso de uso CU-19. Aprobar Pago 83
3.44. Diagrama de Caso de uso CU-06 Gestión de áreas comúes 83
3.45. Diagrama de Actividad Caso de uso CU-20. Listar áreas comúnes . . 85
3.46. Diagrama de Actividad Caso de uso CU-21. Ver área común 86
3.47. Diagrama de Actividad Caso de uso CU-22. Reservar áreas común . . 86
3.48. Diagrama de Secuencia Caso de uso CU-20. Listar áreas comúnes . . 87
3.49. Diagrama de Secuencia Caso de uso CU-21. Ver área común 87
3.50. Diagrama de Secuencia Caso de uso CU-22. Reservar áreas común . . 88
3.55. Diagrama UML del esquema físico de la base de datos. 92
3.51. Diagrama entidad-relación-extendido de la base de datos (proceso 1). 94

3.52. Diagrama entidad-relación-extendido de la base de datos (proceso 2). 95
3.53. Diagrama entidad-relación-extendido de la base de datos (proceso 3). 96
3.54. Diagrama entidad-relación-extendido de la base de datos (proceso 4). 97

4.1. Pruebas en Android Emulator. Cuenta Administrador. 105
4.4. Pruebas automatizadas en la aplicación. 106
4.2. Pruebas en dispositivo físico Galaxys A30. 107
4.3. Pruebas en dispositivo físico Galaxys A30. 108
4.5. Pruebas manuales en Postman.Conguración de la petición. 109
4.6. Pruebas manuales en Postman. Respuesta enviada por el API REST. 109
4.7. Pruebas manuales en Postman. Script de prueba. 110
4.8. Pruebas manuales en Postman.Conguración de la petición. 110
4.9. Pruebas manuales en Postman. Respuesta enviada por el API REST. 111
4.10. Pruebas manuales en Postman. Script de prueba. 111
4.11. Pruebas automatizadas en Postman. Script de prueba. 112

Índice de cuadros

2.1. Principales Características de Aplicaciones para la gestion de condominios 26

3.1. Descripción textual del rol Usuario no Autenticado. 48
3.2. Descripción textual del rol Administrador. 49
3.3. Descripción textual del rol Propietario. 49
3.4. Descripción textual del rol Vigilante. 49
3.5. Descripción del Caso de Uso Crear Cuenta. 50
3.6. Descripción del Caso de Uso Iniciar sesión. 51
3.7. Descripción del Caso de Uso Recuperar Contraseña. 52
3.8. Descripción del Caso de Uso Editar perfil. 57
3.9. Descripción del Caso de Uso Cambiar contraseña. 58
3.10. Descripción del Caso de Uso Cerrar sesión. 59
3.11. Descripción del Caso de Uso Crear condominio. 63
3.12. Descripción del Caso de Uso Buscar condominio. 63
3.13. Descripción del Caso de Uso Ver condominio. 64
3.14. Descripción del Caso de Uso Editar condominio. 65
3.15. Descripción del Caso de Uso Eliminar condominio. 65
3.16. Descripción del Caso de Uso Crear Ingresos/Egresos. 70
3.17. Descripción del Caso de Uso Listar Ingresos/Egresos. 71
3.18. Descripción del Caso de Uso Editar Ingreso/Egreso. 72
3.19. Descripción del Caso de Uso Eliminar Ingreso/Egreso. 72
3.20. Descripción del Caso de Uso Listar Pagos Pendientes. 75
3.21. Descripción del Caso de Uso Ver Pagos. 76
3.22. Descripción del Caso de Uso Enviar comprobante de pago. 76
3.23. Descripción del Caso de Uso Enviar comprobante de pago. 77

3.24. Descripción del Caso de Uso Listar áreas comúes. 84
3.25. Descripción del Caso de Uso Ver áreas comúes. 84
3.26. Descripción del Caso de Uso Reservar área común. 84

4.1. Rutas para operaciones de usuarios 102
4.2. Rutas para operaciones de comndominios 103
4.3. Rutas para operaciones de área común 103
4.4. Rutas para operaciones de noticias 103
4.5. Rutas para operaciones de tarea de mantenimiento 103
4.6. Rutas para operaciones de pagos 104
4.7. Rutas para operaciones de paquetes 104
4.8. Rutas para operaciones de visitas 104
4.9. Rutas para operaciones de gastos 104
4.10. Características del dispotivo fisíco Galaxys A30. 106

Capítulo 1

Introducción

En la actualidad las aplicaciones móviles son altamente utilizadas, ello producto de las facilidades de acceso a la web que proporcionan, así como támbien a las innovaciones tecnológicas de los Smartphone, los cuales hacen uso de sistemas operativos que favorecen el desarrollo de herramientas web sin costos, que son instaladas en un dispositivo móvil sin dificultad alguna (Pedro Illaisaca, 2022).

Los dispositivos móviles son el principal punto de acceso a contenidos e información personal de los usuarios finales: Preveer información relevante, de manera adecuada y proteger la privacidad de sus usuarios, son características únicas de estos tipos de dispositivos.(Moreno, 2022)

Las aplicaciones móviles son herramientas tecnológicas que permiten a las empresas digitalizar los procesos que se ejecutan manualmente, son capaces de aportar soluciones en todos los ámbitos de la sociedad, tales como el turismo, la salud o el comercio electrónico, entre muchos otros sectores. Las Apps han ayudado a mejorar muchos aspectos de la vida cotidiana, por ejemplo, el acceso a la información, el almacenamiento de datos personales de forma segura o la compra de manera inmediata, sencilla y desde cualquier lugar.

El propósito de este proyecto es realizar el diseño e implementación de una aplicación móvil en entorno Android que apoye la gestión administrativa de condominios, permintiendo agilizar en parte sus procesos, logrando un acceso rápido y estructurado a la información tanto para el administrador como para los propietarios.

1.1. Antecedentes

En la actualidad nuestra vida está marcada por un antes y un después de los Smartphone (teléfono inteligente), esta tecnología avanza y evoluciona cada vez más en nuestra sociedad, es la que nos ofrece la posibilidad hoy en día de poder tener todo controlado por medio de estos prácticos dispositivos electrónicos que permiten al usuario realizar tareas como puede ser: gestionar el correo electrónico, cargar páginas Web, e incluso tener todos los datos que necesitemos para trabajar a través de ellos.(Rodriguez, 2020)

(Moreno, 2022), crean una aplicación móvil desarrollada para el sistema operativo Android, que permite el acceso sencillo y rápido a una base de datos mediante códigos QR para consulta de información general del camposanto y específica de los difuntos que allí se encuentran. Se trabaja con un enfoque analítico sintético en la investigación y bajo la metodología MobileD que, de manera ágil, permite el desarrollo de un producto de software de calidad con el soporte de las herramientas Android Studio, My SQL y PHP.

(Pedro Illaisaca, 2022), diseñan una aplicación móvil y web para la insepcción de juntas de agua potable usando servicios de geolocalización y almacenamiento en la nube. Utilizarón la plataforma de desarrollo Ionic que permitió crear una aplicación hibrida, la plataforma de almacenamiento en la nube se implemento en Firebase. Para el desarrollo de la app se utilizó la metología scrum.

(Muñoz, 2021), durante la etapa de desarrollo de la aplicación móvil se empleó la metodología ágil Mobile-D y el framework hibrido Ionic. Indica que la metodología Mobile-D es la mejor elección para organizaciones con equipos muy pequeños, porque al momento de desarrollar una aplicación móvil, esta cumple con un ciclo de desarrollo ágil y también interactúan con el cliente permitiéndole tener un rol importante en el desarrollo de la aplicación. Además realizan una comparación con otras metodologías ágiles que existen en la actualidad como es el caso de las metodologías SCRUM y Programación Extrema (XP).

(Ortiz, 2020) en su proyecto de grado titulado "Desarrollo e implementación de una aplicación móvil informativa para los estudiantes del instituto superior tecnológico primero de mayo", afirma que el uso de metodologías ágiles como Mobile-D son exclusivamente para desarrollo de aplicaciones móviles, esta permite responder rápida-

mente a los cambios que se puedan dar en la etapa de desarrollo del proyecto, lo que permite la reducción del tiempo de producción.

Actualmente en Google Play existen diferentes plataformas que ayudan a mejorar la gestión administrativa de condominios, tales como:

Condomisoft: es un sistema en línea que te permite controlar todo lo relacionado con la administraión de un condominio. Entre sus funcionalidades tiene un blog de noticias, envíos de correo, mensajes privados, información de tareas en procesos, etc.

Residentia: es una app de control de visitantes y administración de condominios y fraccionamientos. Su principal característica es poder registrar visitas programadas, donde el propietario puede generar mediante la app claves de acceso para sus visitantes las cuales se deben proporcionar en vigilancia para que se les permita entrar.

CondoVive: es una app para acceder a los datos del condominio. Entre sus módulos se pueden revisar el estado de cuenta, pagos recientes, reservar áreas comunes, consultar anuncios y enviar comprobantes de pago.

Condominio Organizado: a través de este sistema 100 % en línea, es posible enviar avisos a los propietarios, programar reuniones, reservas espacios comunes, realiza encuestas, controlar acceso. Entre sus móodulos contienen el entega de pedidos con el cual pueden recibir nottificaciones cuando reciban correspondencia.

1.2. Planteamiento del Problema

Un condominio se podría definir como un conjunto de viviendas poseído en régimen de comunidad de propietarios, donde se comparten ciertos gastos en comunes. La administración de algunos condominios es principalmente gestionada por una persona en donde toda la gestión administrativa se lleva a cabo manualmente.

En la administración de condominios se requiere centralizar la información, para lograr una comunicación clara entre los administradores y propietarios. Todos los procesos en que ambos intervienen, como los son las noticias, eventos,pagos y demás deben tener un soporte digital para mejorar la eficiencia de la gestión administrativa.

Actualmente, existen condominios que no cuentan con una aplicación móvil que permita automatizar sus procesos lo cual tiene como consecuencia una mayor capacidad de tiempo, eventuales pérdidas de información y un control de pagos

ineficientes.

1.3. Justificación del Proyecto

La tendencia de usar aplicaciones móviles aumentó significativamente debido a la situación generada por el virus COVID-19, dichas aplicaciones facilitan a las personas consultar y almacenar información en tiempo real sin tener ningún contacto físico.

Se tiene la necesidad de una aplicación móvil que logre mejorar la eficiencia del trabajo de la administración de un condominio creando un acceso rápido y una completa estructuración de la información tanto para el administrador como para los propietarios. Esto ayudaría a mantener el distanciamiento social y la comunicación entre ambas partes, támbien se reduciría los costos de impresión de facturas, documentos, recibos, entre otros.

1.4. Objetivos del Proyecto

El objetivo general de este trabajo, es desarrollar una aplicación móvil que permita automatizar la gestión administrativa de condominios. Para llevar esto a cabo, se debe cumplir con los siguientes objetivos específicos:

1. Revisar de aplicaciones móviles similares y otros recursos que permitan el desarrollo del proyecto.

2. Analizar la gestión y procesos que ocurren en la administración de condominios.

3. Definir los requerimientos funcionales para el desarrollo de la aplicación móvil.

4. Elaborar diseños de los componentes de la aplicación móvil.

5. Desarrollar los módulos de los componentes de la aplicación móvil.

1.5. Metodología

La metodología utilizada para el desarrollo de la aplicación móvil es Mobile-D, metodología ágil que no exige un nivel alto de documentación permitiendo al desarrollador enfocarse más en la creación de la aplicación y terminar en un menor tiempo gracias a la codificación y pruebas que van de la mano.

Mobile-D es una metodología cuyo enfoque y características la hacen especialmente apta para el mercado de dispositivos móviles (Ortiz, 2020) , ya que el desarrollo de una aplicación móvil requiere un enfoque diferente a la construcción de plataformas web. Esto se debe a las características propias de software y hardware presentes en los Smartphone. Su principal objetivo es conseguir ciclos de desarrollo rápidos obteniendo una aplicación basada en módulos que van acorde a las necesidades de los usuarios.

Las fases que se ejecutan en la metodologia Mobile-D se describen en (Muñoz, 2021) como:

Exploración: Esta fase se centra en la planificación y en obtención de los requisitos del proyecto, esto con el fin de tener una visión de la necesidad que el cliente pretender solucionar en la construcción de la solución tecnológica, mediante el establecimiento de las funcionalidades que tendrá el producto.

Inicialización: El objetivo de esta fase es preparar, verificar y configurar todos los recursos que sean necesarios para el desarrollo del proyecto. Cabe resaltar que esta fase se divide en cuatro etapas que son: puesta en marcha del proyecto, planificación inicial, día de prueba y día de salida.

Producción: En esta fase se desarrolla un ciclo de tres días de programación que se repite hasta completar e implementar las funcionalidades proyectadas. Dicho ciclo esta compuesto por: día de planeación, día de trabajo y día de liberación.

Estabilización: En esta fase se llevan a cabo actividades de integración y se verifica el correcto funcionamiento de la solución tecnológica que se esta desarrollando. Cabe

mencionar que en esta fase también se puede incluir la documentación generada en el desarrollo del proyecto.

Pruebas: Una vez se cuente con un desarrollo estable se comienza con un testeo para verificar que la solución tecnológica sea estable y este según lo definido en los requerimientos por parte del cliente. Si es encontrado algún error se repara, pero no se desarrolla nada nuevo ya que haría romper todo el ciclo planteado por la metodología.

Figura 1.1: Metodología MOBIL-D

1.6. Alcances

El alcance de este proyecto es desarrollar los módulos de una aplicación móvil en entorno Android que permita consultar toda la información manejada por la gestión administrativa y así lograr una mejora de la misma.

La aplicación móvil ayudara a la gestión administrativa, en los siguientes módulos y funcionalidades: registro de propietarios, actualización de propietarios, actualización de administrador, envió de correo, noticias, eventos, pagos, reserva de espacios comunes, entre otros.

Capítulo 2

Marco Teórico

En este capítulo se detalla toda la información teórica necesaria para el desarrollo de la aplicación propuesta. En la sección 2.1 se describe todo lo relacionado con la administración de condominios según la Ley de Propiedad Horizontal (LPH) donde se establecen las funciones del administrador, el uso de áreas comunes ademas de definir las cuotas fijas por propietarios, asimismo se realizo una revisión de las aplicaciones exitentes. En la siguiente sección (sección 2.2) se hace una descripción sobre la teoría de desarrollo de aplicaciones móviles . En las secciones 2.3 contiene información de referencia sobre las distintas herramientas tecnológicas utilizadas para el desarrollo.

2.1. Condominios y Administración

Se entiende como condominio o propiedad horizontal el conjunto de apartamentos o locales que residen dentro de un mismo predio susceptibles de aprovechamiento independiente; donde cada persona obtendrá su inmueble ya sea al contado o crédito hipotecario de acuerdo con las disposiciones de la Ley de Propiedad Horizontal (LPH), que tenga salida a la via publica directamente o a través de un determinado espacio común el cual deberá ser conservado por cada uno de los propietarios.

Es importante destacar, que la gestión de un condominio estará regida por la LPH(de la Repúplica, 1983) la cual rige el aspecto formal y jurídico pero adicional a este está el Documento de Condominio donde estará contenido la vida del inmueble, los derechos y obligaciones de cada propietario, definición y distribución de las áreas

comunes; dicho esto, según lo estipulado en la LPH las áreas comunes del inmuebles son:

- La totalidad del terreno que sirvió de base para la obtención del correspondiente permiso de construcción.
- Los cimientos, paredes maestras, estructuras, techos, galerías, vestíbulos, escaleras, ascensores y vías de entrada, salida y comunicaciones.
- Las azoteas, patios o jardines.
- Los locales destinados a la administracón, vigilancia o alojamiento de porteros o encargados del inmueble.
- Los locales e instalaciones de servicios centrales como electricidad, luz, gas, agua fría y caliente, refrigeración, cisterna, tanques y bombas de agua y demás similares.

- Los incineradores de residuos y, en general todos los artefactos, instalaciones y equipos existentes para el beneficio común.
- Los puestos de estacionamiento que sean declarados como tales en el documento de condominio.
- Los maleteros y depósitos en general que sean declarados como tales en el documento de condominio.
- Cualesquiera otras partes del inmueble necesarias para la existencia, seguridad, condiciones higiénicas y conservación del inmueble o para permitir el uso y goce de todos y cada uno de los apartamentos y locales.
- Serán asimismo cosas comunes a todos los apartamentos y locales, las que expresamente se indiquen como tales en el documento de condominio, y en particular los apartamentos, locales, sótanos, depósitos, maleteros o estacionamientos rentables, si los hubiere, cuyos frutos se destinen al pago total o parcial de los gastos comunes.

Una vez determinado las áreas comunes del inmueble cada propietario se le atribuirá una cuota fija de participación; por acuerdo unánime, con relación al total del valor del inmueble y servirá de módulo para determinar la participación en las cargas y beneficios; y así cada uno podrá servirse de dichas áreas según su destino ordinario y sin perjuicio del uso legitimo de los demás. Los apartamentos y locales estarán sometidos a cumplir normas estipuladas en la LPH; entre ellas los propietarios deberán consentir las reparaciones que exija el servicio del edificio y permitir servidumbres imprescindibles requeridas para la creación de servicios comunes de interés general, acordadas por el setenta y cinco (75 %) de los propietarios.

Según lo descrito en la LPH la administración del inmueble estará dividida en la Asamblea General de Copropietarios, la Junta de Condominio y el Administrador. La Junta de Condominio será asignada por la Asamblea de Copropietarios y deberá estar integrada por tres copropietarios y tres suplentes con un lapso de gestión de un (1) año en ejercicio de sus funciones; y será de obligatorio funcionamiento de todos los edificios regulados por dicha ley. La Junta de Condominio decidirá por mayoría de votos y tendrá las atribuciones sobre vigilancia y control sobre la Administración que establezca el Reglamento de la presente Ley y, en todo caso tendrá las siguientes:

- Convocar en caso de urgencia a la Asamblea de Copropietarios.
- Proponer a la Asamblea de Copropietarios la destitución del Administrador.
- Ejercer las funciones del Administrador en caso de que la Asamblea de Copropietarios no hubiere procedido a designarlo
- Velar por el uso que se haga de las cosas comunes y adoptar la reglamentación que fuere necesaria
- Velar por el correcto manejo de los fondos por parte del Administrador.

En el caso del Administrador, será designado en la Asamblea de Copropietarios por mayoría de votos por un periodo de un (1) año. Entre las funciones correspondientes al Administrador se encuentran:

- Cuidar y vigilar las áreas comunes.

- Realizar o hacer realizar los actos urgentes de administración y conservación, así como las reparaciones menores de las áreas comunes.
- Cumplir y velar por el cumplimiento de las disposiciones del documento de condominio, de su reglamento y de los acuerdos de los propietarios
- Recaudar de los propietarios lo que a cada uno le corresponda en los gastos y expensas comunes y si hubiere apartamentos rentables propiedad de la comunidad recibir los cánones de arrendamiento y aplicarlos a los gastos comunes; en caso de que lo recaudado supere a los gastos comunes, los propietarios por mayoría, podrán darle un destino diferente u ordenar su distribución
- Ejercer en juicio la representación de los propietarios en los asuntos concernientes a la administración de las áreas comunes. Para ejercer esta facultad deberá estar debidamente autorizado por la Junta de Condominio, de acuerdo con lo establecido en el respectivo documento. Esta autorización deberá constar en el Libro de Actas de la Junta de Condominio.
- Llevar la contabilidad de los ingresos y gastos afecten al inmueble y a su administración, en forma ordenada y con la especificación necesaria, así como conservar los comprobantes respectivos, los cuales deberán ponerse a la disposición de los propietarios para su examen durante días y horas fijadas con conocimiento de ellos.

2.1.1. Aplicaciones para la administración de condominios

Actualmente, para un mejor manejo, gestion e informacion de los recursos del condominio se ha optado por emplear aplicaciones moviles; por esta razon se abordaran las caracteristicas principales de algunas aplicaciones relacionadas con la gestion administrativa de condominios con el fin de comparar dichas aplicaciones exitentes con la aplicacion a desarrollar.

- **Condomisoft** [1] **:** Es una aplicación desarrollada en México que permite controlar todo lo relacionado con la administración de condominios ideal tanto para

[1] https://apps.apple.com/co/app/condomisoft/id1502494496

residentes que administran su propio condominio o para profesionales que gestionan varios condominios. Este sistema proporciona tres modalidades de uso; la primera modalidad donde únicamente los administradores tienen el acceso al sistema, la segunda tanto administradores como propietarios pueden acceder; y finalmente, la tercera modalidad pueden crear una integración con un sistema web.

- **Residentia** [2]**:** Es un software de Administración de Condominios compatible con sistemas operativos Android y iOS, que sirve como herramienta tanto de los administradores como para los habitantes de condominios, fraccionamientos privadas, edificios residenciales o conjuntos cerrados que pagan mensualmente una cuota de mantenimiento para cubrir necesidades comunes. Esta aplicación ayuda a los propietarios a mantenerse informados de todo lo que ocurre en su condominio, mediante la publicación de comunicados, avisos por parte del Administrador, así como solicitudes por parte de los condominios. Asimismo, integra mecanismos eficaces para que el administrador o tesorero tengan el control tanto de los pagos de mantenimiento realizados por los propietarios, como los gastos efectuados mes con mes.

- **CondoVive** [3]**:** Es una aplicación que maneja múltiples funciones del condominio con facilidad para administradores, la mesa directiva y los propietarios. Además, la aplicación cuenta con dos interfases diferentes, una para el administrador donde puede visualizar las opciones de un programa contable de presupuestos, avisos y reportes, mientras que para el propietario cuenta con una interfaz sencilla y diseñada específicamente para ellos.

- **Condo Control** [4]**:** Es una aplicación móvil que permite manejar cuatro tipos de roles; administradores de propiedades, miembros de la junta, guardias de seguridad y propietarios. Entre sus funciones principales esta el registro de visitantes.Ya sea que esté buscando reservar la sala de fiestas, reservar un lugar de estacionamiento para visitantes.

[2]https://play.google.com/store/apps/details?id=mx.residentia.v2
[3]https://play.google.com/store/apps/details?id=com.ionicframework.condoviveapp
[4]https://play.google.com/store/apps/details?id=com.condocontrolcentral.cccapp

- **Edipro**[5]**:** Es una aplicación móvil desarrollada en Chile para el manejo de la administración de condominios, la principal funcionalidad que ofrecen es poder contratar un administrador certificados en dicha aplicación. Ademas facilitan la opción de seguimiento de paquetes atraves de envío de notificaciones al momento de ser entregado y recibido.

2.1.2. Prinicipales características de la aplicación propuesta

A continuación, se describen las principales funcionalidades de la aplicación a desarrollar (hipotesis planteada)

- Manejará tres tipos de roles de acceso; estos son administrador, propietarios y vigilante. El primero representa al administrador que sera el encargado de crear lo correspondiente al funcionamiento del condominio. El segundo esta formado por los propietarios; los cuales podrán consultar toda la información sobre el condominio. Finalmente el tercero representa a la vigilancia; ellos podrán hacer el registro de paquetes y visitas.

- La aplicación se dividirá en cuatros (4) secciones:

1. **Contabilidad:** En el área de contabilidad se gestionará lo referente a los pagos de gastos comunes; registrar cuentas bancarias del condominio, registrar los pagos realizados, registrar recargos de penalizaciones por falta de pagos.
2. **Comunicación:** En esta área se manejará todas las funciones correspondiente a la comunicación entre administradores y propietarios; se podrá enviar mensajes mediante envíos de notificaciones push, un blog de noticias. Así mismo, se podrá crear y gestionar el estado de las tareas de mantenimientos realizadas dentro del condominio.
3. **Vigilancia:** En esta sección manejaran los procesos de registro de paquetes, registro visitantes.
4. **Otras funciones:** reserva de areas comunes.

[5]https://play.google.com/store/apps/details?id=com.condocontrolcentral.cccapp

- La aplicacion permitira la opción que un administrador pueda gestionar varios condominios.
- El registro de visita se manejara mediante codigo QR.

2.1.3. Características Diferenciadoras de la Aplicación Propuesta con las Aplicaciones Revisadas

En las características descritas previamente, se puede observar que la aplicación propuesta contiene características similares a las apliaciones exitentes; sin embargo se harán algunas comparaciones para detallar las diferencias que exiten entre estás:

- Las aplicaciones existentes en el mercado comparten ciertas características; sin embargo ninguna de estás logra unificarlas. Con la aplicación propuesta se pretende unir todas las funciones que se detallan en la Tabla 4.2.2 en una sola; con el fin de obtener un óptimo manejo de la administración de condominios.
- Actualmente en Venezuela las aplicaciones para la administración de condominios son escasas; por esta razón la aplicación propuesta va dirigida al mercado Venezolano, permitiendo un mayor alcance en los condominios o inmobiliarias que se dediquen administrar condominios.

2.2. Desarrollo de Aplicaciones Móviles:

Las tecnologías móviles están adquiriendo una gran importancia en el campo del desarrollo de software ya que en el mercado actual podemos encontrar una gran variedad de dispositivos compuestos por sistemas operativos y hardware que ofrecen al usuario una mejor experiencia.

Al empezar el desarrollo de una aplicación móvil es importante considerar factores como el ecosistema móvil, las metodologías que se van aplicar hasta el mercado que va dirigido la aplicación; tener en cuenta estos factores aseguran un desarrollo exitoso.

Tabla 2.1: Principales Características de Aplicaciones para la gestion de condominios

Condomisoft	▪ Registro Ingresos y Egresos. ▪ Pagos en Línea. ▪ Recordatorios de pagos. ▪ Blog General. ▪ Registro de gastos comunes.
Residentia	▪ Control de Visitas. ▪ Encuestas y Votaciones. ▪ Registro de ingresos y egresos. ▪ Reservación de áreas comunes. ▪ Registro de gastos comunes.
CondoVive	▪ Manejar múltiples condominios. ▪ Encuestas y Votaciones. ▪ Registro de ingresos y egresos. ▪ Reservación de áreas comunes. ▪ Registro de gastos comunes.
Condo Control	▪ Seguimiento de tareas. ▪ Anuncios y boletines. ▪ Pagos en Línea. ▪ Reservaciones de áreas comunes. ▪ Control de visitas. ▪ Registro de gastos comunes.
Edipro	▪ Control de Visitas. ▪ Mesajería. ▪ Registro de ingresos y egresos. ▪ Reservaciones de áreas comunes. ▪ Registro de gastos comunes.
CondoAgil	▪ Manejar múltiples condominios. ▪ Control de Visitas. ▪ Mesajería. ▪ Registro de ingresos y egresos. ▪ Reservaciones de áreas comunes. ▪ Registro de gastos comunes. ▪ Seguimiento de tareas. ▪ Seguimiento de paquetes.

Por ejemplo, las metodologías de desarrollo a utilizar en el ámbito móvil, difieren de las metodologías usadas en las aplicaciones de escritorio o web. En las aplicaciones móviles se deben tomar en cuenta factores como el consumo de batería, gestos, orientación, conectividad, teclado, API, factores que para las aplicaciones de escritorio no son un mayor problema. Por otro lado las aplicaciones móviles requieren ser actualizadas en periodos de tiempos más cortos que los de las aplicaciones tradicionales, para solventar ciertos problemas que puedan ocurrir y seguir ofreciendo más y mejores características que permitan mejorar la experiencia del usuario (UX, por sus siglas en inglés) satisfaciendo sus gustos y necesidades. (Boscán, 2020)

2.2.1. Ecosistema móvil

Por ecosistema móvil nos referimos al conjunto de actores necesarios para poder tener los dispositivos móviles y las aplicaciones para los mismos. En concreto, en el ecosistema móvil se incluyen las operadoras de telecomunicaciones, los fabricantes de hardware y todos los elementos de software que intervienen en la ejecución de la aplicación.(Vique, 2018)

Todas las aplicaciones se ejecutan dentro de un ecosistema; estos ecosistemas constituyen una de las mayores dificultades en lo que respecta al desarrollo de aplicaciones ya que acaba causando, entre otras cosas, una mayor fragmentación de la aplicación. Mediante el analisis del ecosistema se podra obtener desde la información de la red de datos actual para adaptar los contenidos hasta la información del propio dispositivo, ademas encontramos capacidades que en otros entornos no encontraríamos, como la de localizar otros dispositivos en movimiento, la de dar información sobre nuestro entorno (localización, orientación, presión atmosférica, etc.), la de conseguir información sobre el usuario (contactos, calendario, etc.) o, incluso, la de disponer de medios de pago mucho más directos (mediante la operadora o mediante el propio dispositivo).

En (Boscán, 2020) se listan los actores a tener en cuenta en el ecosistema de aplicaciones móviles:

- **Servicios:** Son los servicios que ofrecerá la aplicación (servicios de mensajería, compras en linea, streaming, feed de noticias, ...).

- **Aplicaciones:** El tipo de aplicación a desarrollar; pueden ser nativa, hibrida,web
- **Herramientas:** Son los frameworks, IDE y herramientas en general necesarios para construir/desarrollar la aplicación.
- **Sistemas operativos:** El o los sistemas operativos con los que sería compatible la aplicación.
- **Plataformas:** Son todo el conjunto de elementos de software y hardware necesarios para ejecutar la aplicación y con los cuales sera compatible.
- **Dispositivos:** Los dispositivos móviles compatibles para la aplicación. Existen distintos tipos de dispositivos móviles teléfonos inteligentes, tabletas, laptops, relojes inteligentes, entre otros; y dentro de un mismo tipo existen varios dispositivos móviles, en el caso de los teléfonos inteligentes, se tienen Samsung, Huawei, IPhone, IPod, LG, Nokia, entre otros.
- **Redes:** Los tipos de redes móviles necesarios para que la aplicación funcione (EDGE, 3G, 4G, etc.)
- **Sistemas operativos:** El o los sistemas operativos con los que será compatible la aplicación.
- **Usuarios:** Son los usuarios a los que esta dirigida la aplicación, es decir los usuarios que harán uso de la aplicación.

2.2.2. Fragmentación

La fragmentación es una situación, o el conjunto de condicionantes de una situación, en la que no es posible compartir una misma aplicación entre diferentes ecosistemas. Es decir, la fragmentación impide que se pueda compartir la aplicación sin adaptar los ecosistemas; puede ocurrir por muchos factores, los cuales provocan diversidad y entropía en las aplicaciones que queremos crear.

La fragmentación puede originarse por los siguientes motivos:

- **Hardwarer diferente:** Como, por ejemplo, dispositivos con componentes distintos: tamaño o densidad de la pantalla, teclado, sensores, capacidad de proceso, etc.

- **Software diferente:**
  - **Plataforma diferente:** Una plataforma, un framework, un sistema operativo (o las versiones de cualquiera de ellos) puede generar la fragmentación de las aplicaciones.

    Diferencias en las implementaciones: Por ejemplo, diferencias en la implementación del estándar, o bien errores conocidos de versiones concretas.

- **Variaciones de las funcionalidades:** Por ser una versión con menos privilegios (versión de pago y versión gratuita) o según los roles de los propios usuarios de la aplicación.

- **Preferencias de usuario:** Las más habituales son las localizaciones de la aplicación (idioma, orientación del texto, etc.).

- **Diversidad del entorno.:** Derivado de la infraestructura, como pueden ser los operadores y sus API, los problemas de cortafuegos, las limitaciones de las redes, el roaming, etc.

La fragementación puede afectar el desarrollo del proyecto, desde el modelo de negocio hasta el despliegue, además de la implementación y las pruebas. Por esta razón, existen diferentes estrategias para abordar la fragmentación:

- **Un desarrollo para cada escenario:** Se realiza un desarrollo para cada fragementación que se encuentre al hacer el estudio del ecosistema.El proceso de adaptar la aplicación a un nuevo escenario se llama portar la aplicación. Este tipo de estrategia suelen ser muy costosa pues no se puede aprovechar nada (o casi nada) sin adaptaciones del código realizado en otros escenarios. Sin embargo es muy útil para casos en que los escenarios son muy diferentes.

- **Parte común y derivaciones:** Según esta estrategia, una parte de nuestra aplicación es común en todos los escenarios, y para cada uno de ellos se realiza un desarrollo. Al poder aprovechar parte del código realizado se logra disminuir el costo de producción, por esta razón esta estrategia se considera menos costosa que la anterior.No obstante, se requiere conocer tanto los lenguajes o entornos

de los diferentes dispositivos como las herramientas necesarias para llevar a cabo las adaptaciones.

- **Adaptación única:** Mediante está estrategia podemos conseguir que con un sola aplicación funcione en todos los casos sin necesidad de realizar más cambios. A pesar de ser considerada una de las estrategias más económicas suelen utilizarse para aplicaciones simples donde no es necesario acceder a las capacidades o controladores de los dispositivos.

2.3. Herramientas y tecnología de desarrollo:

Esta sección se detallan las principales herramientas y tecnologías utilizadas para desarrollar la aplicación m óvil y el API REST.

2.3.1. Ionic

Ionic es considerado un framework de desarrollo de aplicaciones móviles HTML5 destinado a crear aplicaciones móviles híbridas. Las aplicaciones híbridas son esencialmente sitios web pequeños que se ejecutan en un shell de navegador en una aplicación que tiene acceso a la capa de plataforma nativa. (IonicFramework)

Las aplicaciones híbridas tienen muchos beneficios sobre las aplicaciones nativas puras, específicamente en términos de soporte de plataforma, velocidad de desarrollo y acceso a código de terceros.

Siendo Ionic un marco de interfaz de usuario de FrontEnd que maneja todas las interacciones de interfaz de usuario y apariencia que la aplicación necesita con soporte para una amplia gama de componentes móviles nativos comunes, animaciones elegantes y diseño único; que obtendría con un SDK nativo en iOS o Android, también brinda algunas formas sencillas pero poderosas de crear aplicaciones móviles que eclipsan los marcos de desarrollo HTML5 existentes.

Dado que Ionic es un marco HTML5, necesita un contenedor nativo como Cordova o PhoneGap para ejecutarse como una aplicación nativa; siendo Cordova el contenedor más apropiado para las aplicaciones.

Es importante destacar que las aplicaciones Ionic no están diseñadas para ejecutarse en una aplicación de navegador móvil como Chrome o Safari, sino en el

shell del navegador de bajo nivel como UIWebView de iOS o WebView de Android, que están envueltos por herramientas como Cordova / PhoneGap.

Entre las caracteríisticas más importantes de Ionic se encuentra que es de código abierto permitiendo así tener licencia de código abierto permisivos que pudiera usarse en aplicaciones comerciales y de código abierto, como al cultivar una comunidad sólida en torno al proyecto sin tener que comprar una licencia comercial.

En resumen, Ionic es solo una página web que se ejecuta en un shell de aplicación nativa permitiendo el uso de cualquier tipo de HTML, CSS y Javascript que se requiera. La única diferencia es que, en lugar de crear un sitio web al que otros se vincularán, se estará creando una aplicación autónoma.

2.3.2. Node

Node.js ieado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones network escalables. (NodeJS)

Características principales de Node.js

- **Velocidad:** Node.js está construido sobre el motor de JavaScript V8 de Google Chrome[6], por eso su biblioteca es muy rápida en la ejecución de código.
- **Sin búfer:** Las aplicaciones de Node.js generan los datos en trozos (chunks), nunca los almacenan en búfer.
- **Asíncrono y controlado por eventos:** Como hemos dicho anteriormente, las APIs de la biblioteca de Node.js son asíncronas, sin bloqueo. Un servidor basado en Node.js no espera que una API devuelva datos. El servidor pasa a la siguiente API después de llamarla, y un mecanismo de notificación de eventos ayuda al servidor a obtener una respuesta de la llamada a la API anterior.
- **Un subproceso escalable:** Node.js utiliza un modelo de un solo subproceso con bucle de eventos. Gracias al mecanismo de eventos, el servidor responde sin bloqueos, como hemos dicho. Esto hace que el servidor sea altamente

[6]https://v8.dev/

escalable comparando con los servidores tradicionales como el Servidor HTTP de Apache.

Es importante resaltar que Node.js es compatible con el estilo arquitectónico MVC (Modelo-Vista-Controlador).

2.3.2.1. Express.js

Express[7] es una librería que utiliza Node.js, que ayuda a manejar de forma sencilla las peticiones HTTP en el servidor. Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Integración con motores de renderización de "vistas"para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware.ªdicional en cualquier punto dentro de la tubería de manejo de la petición.

Una de las ventajas de express es su flexibilidad al momento de configurarlo. No requiere ni obliga a realizar la organización de distintos componentes de ninguna forma en específica. Sin embargo se recomienda estructurarlo de la siguiente forrma:

- **app.js:** Es el archivo principal en el cual se encuentra la lógica del servidor y de la aplicación.
- **/public:** Contiene archivos estáticos que son desplegados por el servidor.
- **/routes:** Contiene rutas personalizadas para API?s basadas en REST.
- **/views:** Contiene plantillas que pueden ser procesadas por un motor de plantillas.
- **/package.json:** Archivo manifest del proyecto.
- **/www:** Contiene los scripts de arranque del servidor.

Express.js fue el framework utilizado para la creación de la API REST.

[7] https://developer.mozilla.org/es/docs/Learn/Server-side

2.3.3. Heroku

Heroku es una plataforma como servicio (PaaS) de computación en la Nube que soporta distintos lenguajes de programación[8]. Ejecuta las aplicaciones del cliente en contenedores virtuales que se ejecutan en un entorno de tiempo real, puede ejecutar código escrito en Node, Ruby, PHP, Go, Scala, Python, Java o Clojure.

Posee una fácil configuración y funciona con el sistema de control de versiones Git, además, puede ser conectado con la plataforma de alojamiento de proyecto GitHub, para que los cambios que sean reflejados en GitHub se sincronicen automáticamente con Heroku.(Boscán, 2020)

Para hospedar la aplicación en un servidor de producción se utulizo Heroku.

2.3.4. Emuladores

Los emuladores son dispositivos virtuales que corren en un computador, tienen como finalidad apoyar en el diseño y depuración de aplicaciones en un ambiente similar al que existe en un dispositivo real. (Cruz et al., 2020); proporcionando características, tanto de hardware, como de software.

El emulador oficial para Android es Android Emulator, siendo un software libre cuenta con un SDK disponible para todo desarrollador que lo desee; además es multiplataforma estando disponible para Windows,MacOs y Linux. El IDE xCode es el emulador oficial para iOS, es un entorno de trabajo diseñado por Apple; dicho entorno ofrece todas las herramientas necesarias para programar tanto para Mac OS X como iOS incluyendo un simulador para diferentes versiones de OS con sus diferentes dispositivos correspondientes.

2.3.4.1. Android Emulator

Android Emulator simula dispositivos Android en una computadora para que puedas probar tu app en diferentes dispositivos y niveles de API de Android sin necesidad de contar con los dispositivos físicos. El emulador proporciona casi todas las funciones de un dispositivo Android real. Puedes simular llamadas y mensajes de texto entrantes, especificar la ubicación del dispositivo, utilizar diferentes velocidades

[8]https://www.heroku.com/

de red, probar sensores de rotación y otros sensores de hardware, acceder a Google Play Store y mucho más.(DeveloperAndroid)

Es importante destacar que existen varias funciones que el emulador no tiene compatible, estas son: Bluetooth,WiFi, USB, inserció/expulsión de tarjetas SD, y auriculares conectados a dispositivos.

Depurar la aplciación en el emulador es más rápido y fścil que hacerlo en un dispositivo físico. Por ejemplo, se pueden transferir datos con mayor velocidad al emulador que a un dispositivo conectado mediante USB. El emulador támbien incluye configuraciones predefinidas para varios teléfonos y tablets Android, Wear OS y dispositivos Android TV.

2.3.5. Postman

Es una herramienta que permite a crear peticiones sobre APIs de una forma muy sencilla y poder de esta manera probar las APIs. Todo basado en una extensión de Google Chrome., el usuario de Postman puede ser un desarrollador que esté comprobando el funcionamiento de una API para desarrollar sobre ella o un operador el cual esté realizando tareas de monitorización sobre un API. (Arquitectoit)

Gracias a la idea de testear las APIs, Postman es considerada una plataforma de desarrollo de APIs que se basa por un modelo de desarrollo API First ofreciendo un conjunto de utilidades adicionales para poder gestionar las APIs proporcionando herramientas para documentar los APIs, realizar una monitorización sobre las APIs y crear equipos sobre un API para que trabajen de forma colaborativa.

Entre las principales características de Postman, se encuentran:

- Crear Peticiones, permitiendo crear y enviar peticiones HTTP a servicios REST mediante un interfaz gráfico. Estas peticiones pueden ser guardadas y reproducidas a posteriormente.

- Definir Colecciones, mediante Postman se pueden agrupar las APIs en colecciones. En estas colecciones, se definen el modelo de autentificación de las APIs para que se añada en cada petición. De igual manera se ejecutan un conjunto de test, así como definir variables para la colección.

- Gestionar la Documentación, genera documentación basada en las API y colecciones que se han creado en la herramienta.
- Entorno Colaborativo, permite compartir las API para un equipo entre varias personas. Para ello se apoya en una herramienta colaborativa en Cloud.
- Genera código de invocación, dado un API es capaz de generar el código de invocación para diferentes lenguajes de programación: C, cURL, C#, Go, Java, JavaScript, NodeJS, Objective-C, PHP, Python, Ruby, Shell, Swift,?
- Establecer variables, Postman permite crear variables locales y globales que posteriormente se utilizaran dentro de las pruebas.
- Soporta Ciclo Vida API management, desde Postman se puede gestionar el ciclo de vida del API Management, desde la conceptualizacón del API, la definición del API, el desarrollo del API y la monitorización y mantenimiento del API.
- Crear mockups, brinda la opción de crear un servidor de mockups o sandbox para que se puedan testear las API antes de que estén desarrolladas.

Esta herramienta se utilizó para probar el API REST construido con Node.js

2.3.6. One signal

OneSignal es la plataforma de notificación Push más ampliamente utilizada en la plataforma web y aplicaciones móviles, admitiendo las principales plataformas nativas y móviles debido a un SDK y API dedicado para cada plataforma. Además, proporciona un panel donde se podrá personalizar las notificaciones Push a enviar. (OneSignal)

Es importante señalar que OneSignal posee una plataforma multicanal que ofrece notificaciones push móviles y web , mensajería en la aplicación , SMS y correo electrónico. Asimismo, al proporcionar una API abierta, documentación extensa, cuentas gratuitas y herramientas intuitivas de personalización y análisis OneSignal se adapta a empresas de todos los niveles a brindar una experiencia de mensajería perfecta. [9]

[9] https://onesignal.com/about

Entre las nuevas funcionalidades de OneSignal :

- **Mensajería en la aplicaciíon de Android e iOS:** Cuando la aplicación esta registrada en OneSignal, se podrán crear mensajes visuales que lleguen a todos los usuarios de dicha aplicación, evitando periodos largos de revisión por parte de la tienda de la aplicación (Google Play Store, App Store), así como ahorro de tiempo y esfuerzo en la creación de código para mensajes internos. Además permite aplicar la segmentación para comprobar qué tipo de usuarios visualizan el mensaje en la aplicación y cuando enviarlos, permitiendo la personalización de los mensajes según las preferencias de los usuarios incluyendo imágenes, que además, podrán tener una durabilidad estimada.

- **Mejora de la API del historial de notificaciones:** Con la mejora de la API, se podrá conocer los usuarios que han visto un mensaje determinado y asi orientar para la creación de nuevas campañas.

One signal fue utilizado para enviar las notificaciones push en la aplicación.

2.3.7. Base de Datos

Las bases de datos se pueden dividir en dos grupos generales:

- **Base de datos no relacionales(NoSQL):**las bases de datos no relacionales son las que, a diferencia de las relacionales, no tienen un identificador que sirva de relación entre un conjunto de datos y otros. Se utilizán principalmente en aplicaciones para videos juegos, sistemas de análisis en tiempo real, etc.

  Entre los gestores más populares encontramos MongoDB seguida por Redis, Elasticsearch y Cassandra.

- **Base de datos relacionales(SQL):** se basan en la organización de la información en trozos pequeños, que se relacionan entre ellos mediante la relación de identificadores o claves primarias.

  Los datos están almacenados en tablas(entidades/relaciones) y cada una de estas es un conjunto de registros (filas y columnas), las tablas se relacionan a través de las claves foráneas.

Para asegurar que los datos se almacenen correctamente se necesita tener claro ciertas reglas o restricciones de validación:

- Integridad de Dominio: Conjunto de valores válidos de un campo (propiedades del campo)
- Integridad de Transiciones: Define los estados por lo que un registro puede pasar válidamente (operación previa)
- Integridad de Entidades: Asegura la integridad de las tablas (claves, identificación)
- Integridad Referencial: Mantienen y protegen vínculos entre tablas (propiedades de las relaciones)
- Integridad de Bases de Datos: Referencian más de una tabla, gobiernan la DB como un todo.
- Integridad de Transacciones: Controlan la forma como se manipulan los datos entre una o varias BD

El lenguaje más común para construir las consultas a bases de datos relacionales es SQL (Structured Query Language), los comando SQL se utilizan tanto para consultas interactivas como para obtener información de una base de datos relacional y la recopilación de datos. Entre los gestores o manejadores actuales más populares encontramos: MySQL, PostgreSQL, Oracle, DB2, INFORMIX, Interbase, FireBird, Sybase y Microsoft SQL Server.

Escoger el tipo de base a utilizar dependerá de las caraacteristtícas y requisitos establecido de la aplicación, para esté desarrollo se opto por la base de datos relacional ya que los datos no crecen tan rápido.

Capítulo 3

Análisis y diseño de la Aplicación

En este capítulo se expone todo lo relacionado con las dos primera fases de la metodología que se eligio anteriormente (Mobile-D). En la primera fase exploración se establecieron los requerimientos necesiarios del proyecto, luego se continuo con la siguiente fase inicialización donde se prepararon todos los recursos necesarios para el desarrollo; se utilizaron diferentes tipos de diagramas para lograr definir el comportamiento y estructura de la aplicación.

3.1. Fase 1: Exploración

En está fase de exploración se establecen las funcionalidades y condiciones que debe cumplir la aplicación; a continuación se detallan las actividades que conforman esta fase que comprenden la definición de los requisitos de software, su clasificación y la personalización del servicio.

Descripción términos:

- **Usuario no registrado:** Es un usuario que desee usar la aplicación pero no tienen cuenta. Puede ser un administrador,propietario o vigilante de un conjunto residencial.

- **Administrador:** Se refiere a un usuario registrado en la aplicación con el rol de Administrador.

- **Propietario:** Se refiere a un usuario registrado en la aplicación con el rol de Propietario.
- **Vigilante:** Se refiere a un usuario registrado en la aplicación con el rol de Vigilante.
- **Usuario Registrado:** Es un usuario con una cuenta creada, puede ser un Administrador, Propietario o Vigilante.
- **Condominio:** El término Condominio se utilizará para identificar el conjunto de dos o mas propietarios que serán administrados por los usuarios con rol Administrador.
- **API REST:** Se utilizaran los términos API REST o API para referirse a lo mismo.
- **Egresos:** El término de egreso se utilizará para referirse a los pagos realizados por el admistrador para cancelar algún servicio solicitado.
- **Ingresos:** El término ingreso se utilizará para referirse a los pagos realizados por el propietario.
- **Gastos comúnes:** Se refiere a las cuotas fijas creadas por el administrador; dichas cuotas serán consideradas como un ingreso.
- **Información de perfil:** Es toda la información (obligatoria u opcional) almacenada en el sistema, perteneciente a un Administrador o Propietario en específico.
- **Información detallada:** Es toda la información sobre un Condominio, Propietario, Vigilante o Administrador.
- **Aplicación móvil:** Se utilizaran los términos aplicacioón móvil o aplicación de manera intercambiable para referirse a lo mismo.
- **Consulta de información por Administrador:** Consulta realizada por los usuarios con rol Administrador donde se consulte la información disponible sobre un condominio, pagos realizados, tareas en curso, entre otros.

- **Consulta de información por Propietario:** Consulta realizada por los usuarios con rol Propietario donde se consulte la información disponible sobre pagos, reserva de áreas comúnes entre otros.
- **Consulta de información por Vigilantes:** Consulta realizada por los usuarios con rol Vigilante donde se consulta la información disponible sobre visitas y paquetes.

3.1.1. Actividad 1: Obtención de los requisitos

Se describen de manera general las características que debe tener la aplicación para solventar una necesidad:

Descripción de la aplicación:
La aplicación funcionará como una plataforma donde un Administrador podrá manejar varios condominios, así mismo tendrá la opción de consultar toda la información relacionada con un condominio en específico, podrán registrar egresos e ingresos, verificar reservas de áreas comunes, registro de tareas de mantenimiento, entre otros. La aplicación también permitirá que los propietario puedan monitorear todas las tareas de mantenimeinto que se están realizando, podrán registrar las visitas que van a recibir; támbien tendrán la opción de registrar los pagos; igualmente los Vigilantes podrán tener un control sobre las visitas y los paquetes correspondiente a cada propietario. Ademas los usuarios vigilantes pondrán registrar las visitas y paquetes. Otra característica que debe tener la aplicación es permitir la comunicación entre administrador y propietarios mediante un chat directo.

Descripción de funcionalidades:

- **Usuario no registrado:**
  - Registrarse bajo el rol de Administrador, Propietario o Vigilante.
  - Iniciar sesión.
- **Administrador:**
  - Crear pagos (cuotas fijas, cuotas especiales y cuotas de penalizaciones).

- Verificar pagos.
- Registrar gastos mensuales.
- Verificar cuenta de vigilantes.
- Edtiar o eliminar cuenta de propietarios.
- Registrar, editar y eliminar tareas de mantenimiento.
- Enivar notificaciones a los propietarios.
- Publicar, editar y eliminar noticias.
- Enviar mensajes a los propietarios
- Consulta de registro de Visitas.
- Consulta de registro paquetes.
- Consulta de áreas comúnes.

- **Propietarios:**

- Registrar pagos.
- Programar y eliminar visitas.
- Listar entrega de paquetes.
- Reservar áreas comúnes.
- Consulta de noticias.
- Consulta de tareas de mantenimiento.

- **Vigilantes:**

- Registrar visitas.
- Registrar paquetes.
- Enviar notificaciones de emegencia a los propietarios.

Información relevante: A continuación se listan la información que debe guardar el sistema para cada una de las entidades:

- **Administrador:**

- Nombre.
- Correo electrónico.
- Número de teléfono.
- ID de administrador.

- **Propietario:**

- Nombre.
- Correo electrónico.
- Código de condominio.
- Número telefónico.
- Número apatarmento.

- **Condominio:**

- Nombre.
- Correo electrónico.
- Logo.
- Descripción.
- Código de condominio.
- Cantidad de apartamentos.
- Número telefónico.
- Dirección del condominio.
- ID de administrador.
- Registro único de información fiscal (R.I.F.).

- **Vigilante:**

- Nombre.
- Correo electrónico.
- Código de condominio.
- Número telefónico.

3.1.2. Actividad 2: Clasificación de los requisitos:

- **Requisitos Funcionales:** Los requerimientos funcionales de un sistema, son aquellos que describen cualquier actividad que la aplicación, estos se detallan a continuación.

Registro y sesiones de usuarios:

RF1. El sistema debe permitir a cualquier usuario registrarse como Administrador, propietario o vigilante.

RF2. Para registrarse como administrador el sistema debe solicitar y almacenar la siguiente información: nombre, apellido, coreo electrónico, contraseña, número telefónico.

RF3. Para registrarse como propietario el sistema debe solicitar y almacenar la siguiente información: nombre, apellido, coreo electrónico, contraseña, número telefónico, número de apartamento, código de condominio.

RF4. El sistema debe permitir al administrador poder registrar a un condominio ingresando la siguiente información: nombre, dirección, cantidad de apartamento, registro único de información fiscal, foto

RF5. El sistema debe permitir a los usuarios registrados recuperar su contraseña de inicio de sesión a través del correo electróonico del usuario.

RF6. El sistema debe permitir a cualquier usuario poder iniciar sesión ingresando su correo electrónico y contraseña.

RF7. El sistema debe permitir a cualquier usuario que haya iniciado sesión poder cerrar su sesión.

RF8. El sistema debe permitir a un administrador poder registrar varios condominios.

Contabilidad:

RF9. El sistema debe permitir a los administradores poder registrar cuentas bancarias por cada condominio.

RF10. El sistema debe permitir a los administradores poder registrar los egresos del condominio.

RF11. El sistema debe permitir a los administradores poder verificar y aprobar los ingresos del condominio.

RF13. El sistema debe permitir a los propietarios poder registrar los pagos realizados.

Comunicación:

RF14. El sistema debe permitir que los administradores puedan comunicarse a través de un chat privado.

RF15. El sistema debe permitir a los administradores poder enviar notificaciones push.

RF16. El sistema debe permitir a los vigilantes poder enviar notificaciones push de alertas de incidentes.

RF17. El sistema debe permitir a los administradores poder crear y eliminar noticias.

RF18. El sistema debe permitir a los propietarios poder consultar noticias.

RF19. El sistema debe permitir a los administradores poder crear, editar y eliminar tareas de mantemiento.

RF20. El sistema debe permitir a los propietario poder consultar las tareas de mantemiento.

Viigilancia:

RF21. El sistema debe permitir a los vigilantes poder consultar las visitas ingresadas por los propietarios.

RF22. El sistema debe permitir a los propietarios poder registrar visitas.

RF23. El sistema debe permitir a los vigilantes poder registrar los paquetes recibidos.

RF24. El sistema debe permitir a los propietarios poder consultar el registro de paquetes.

Otras funciones:

RF25. El sistema debe permitir a los administradores crear áreas comúnes.

RF26. El sistema debe permitir a los propietarios realizar reservaciones de áreas comúnes.

- **Requisitos no Funcionales:** La aplicación debe cumplir con una serie de requisitos no funcionales relativos a su arquitectura, seguridad, uso de estándares, interfaz de usuario.

  Arquitectura

  RNF1. La metodología de desarrollo de software será Mobile-D.
  RNF2. Los datos deberán estar almacenados en una base de datos.
  RNF3. La API REST tendrá el estilos arquitectónico REST.
  RNF4. La aplicación contará con dos partes, frontend(será una aplicación móvil) y backend(será un API REST).

  Seguridad

  RNF5. Los datos de un condominio solo podrá ser modificado por un administrador.
  RNF6. Las contraseñas de los usuarios deben estar cifradas.

  Estándares

  RNF7. Para las pruebas de emulación se usará el emulador oficial de Android (Android Emulator).
  RNF8. Las pruebas del API REST se llevarán a cabo utilizando la herramienta Postman.
  RNF9. Para las pruebas en dispositivos reales se usarán los teléfonos inteligentes Samsung A31 y Xiami Pocco X3.

  Interfaz de usuario

  RNF10. La aplicación debe contar una estructura clara, ordenando el contenido y las funciones en pestañas o apartados que abarquen todas las funcionalidades disponibles, según el rol del usuarsio conectado.
  RNF11. El sistema debe contar con un manual básico de uso.
  RNF12. El sistema debe proporcionar mensajes de error que sean informativos y orientados al usuario final.

- **Requisitos de Entorno:** La aplicación debe cumplir con una serie de requisitos no funcionales relativos a su arquitectura, seguridad, uso de estándares, interfaz

de usuario.

Comunicación cliente-servidor:
RE1. La aplicación debe mantener conxeión constante con el servidor.
RE2. Las solicitudes enviadas por la aplicación al servidor deben utilizar el protocolo HTTP, devolviendo la respuesta en formato JSON.

Tecnologías o herramientas:
RE3. La aplicación móvil debe ser desarrollada utilizando el framework Ionic, junto con el lenguaje de programación TypeScript.
RE4. El API REST debe ser desarrollado utilizando Node.js, con la librería Express.js y el lenguaje de programación Javascript.
RE5. La base de datos será una base de datos relacional y estará implementada en el sistema gestor de base de datos MySQL.
RE1.La base de datos estará almacenada en el servidor.

3.2. Fase 2: Inicialización

En esta etapa detallan todas las actividades correspondiente a la segunda etapa de la metodología, se definen los diagrmas UML para modelar y esquematizar el sistemas desde perceptivas de uso, estructura, interacción, comportamiento e implementación

3.2.1. Actividad 2: Estructuración del Software

Tomando en cuenta los requisitos planteados en la etapa exploración, se definen los actores del sistema; se realiza la construcción de los diagramas de casos de uso, diagramas de actividades y diagramas de secuencias; el diseño conceptual, lógico y físico de la base de datos y el diseño de los esquemas de pantallas.

3.2.1.1. Roles/Actores del sistema

En la figura 3.1 se detallá de manera gráfica los actores que conforman el sistema; en general son 4 roles que puede tomar un usuario, el primero es el usuario no autenticado, que representa los usuarios que no han iniciado sesión. Los roles restantes representan a los usuarios auutenticados y estan conformados por el rol Administrador, propietario y vigilante.

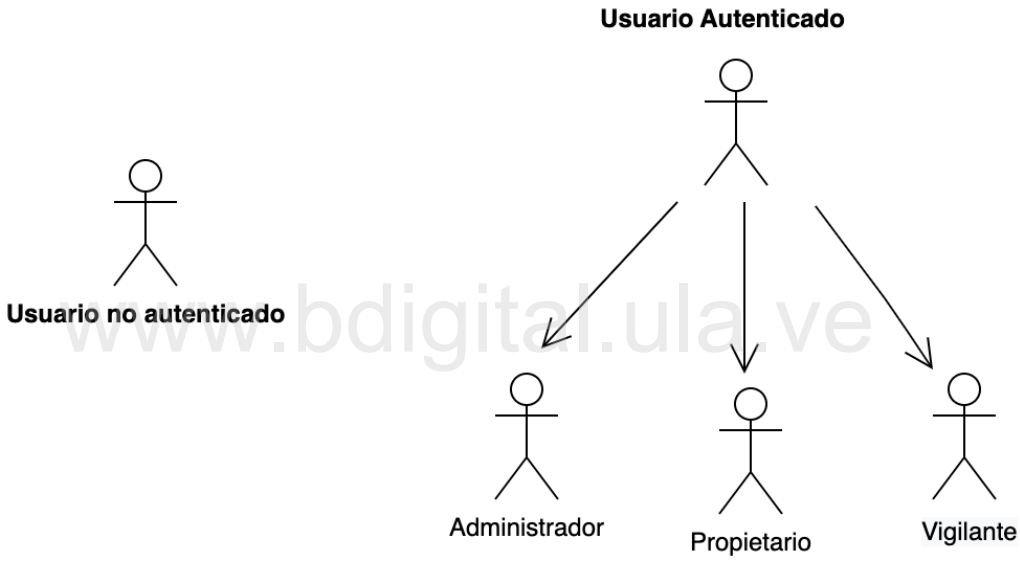


Figura 3.1: Diagrama de Actores.

En las tablas Tabla 3.2.1.1, 3.2.1.1, 3.2.1.1 y 3.2.1.1 se detallan la descripcón de cada rol mostrado en la figura anterior.

Tabla 3.1: Descripción textual del rol Usuario no Autenticado.

Nombre	Usuario no Autenticado
Descripción	Usuarios que no se han registrado en el sistema o usuarios que están registrado pero no no han iniciado sesión.

Tabla 3.2: Descripción textual del rol Administrador.

Nombre	Administrador
Descripción	Usuarios que se registraron en el sistema con rol Administrador; podrá crear y administrar condomios, registrar pagos, cuotas fijas y penalizaciones, crear noticias, enviar mensajes a clientes, entre otras.

Tabla 3.3: Descripción textual del rol Propietario.

Nombre	Propietario
Descripción	Usuarios que se registraron en el sistema con rol Propietario; podrá registrar pagos, consultar noticias, registrar paquetes, reservas áreas comunes, enviar mensajes.

Tabla 3.4: Descripción textual del rol Vigilante.

Nombre	Propietario
Descripción	Usuarios que se registraron en el sistema con rol vigilante; podrá regsitrar visitas, registrar paquetes y enviar alertas a los propietarios.

3.2.2. Casos de uso, diagramas de secuencias y diagramas de actividades del sistema:

En esta sección se describe el funcionamiento del sistema. Se utilizan diagramas de caso de uso para hacer un representación gráfica de las distintas interacciones que tendrán los actores con el sistema; posteriormente se detallan de manera textual cada caso de uso, luego, se realizaron los diagramas de actividades y secuencia de estos casos de uso.

3.2.2.1. Iniciar sesión, crear cuenta y recuperar contraseña:

Los usuarios que no se han registrado en el sistema, pueden crear cuenta y posteriormente iniciar sesión. Los usuarios que están registrado en el sistema

pueden iniciar sesión; en caso de que olviden la contraseña podrán acceder al sistema mediante la recuparción de contraseña. En la 3.2 se resumen las tres operaciones descritas anteriormente.

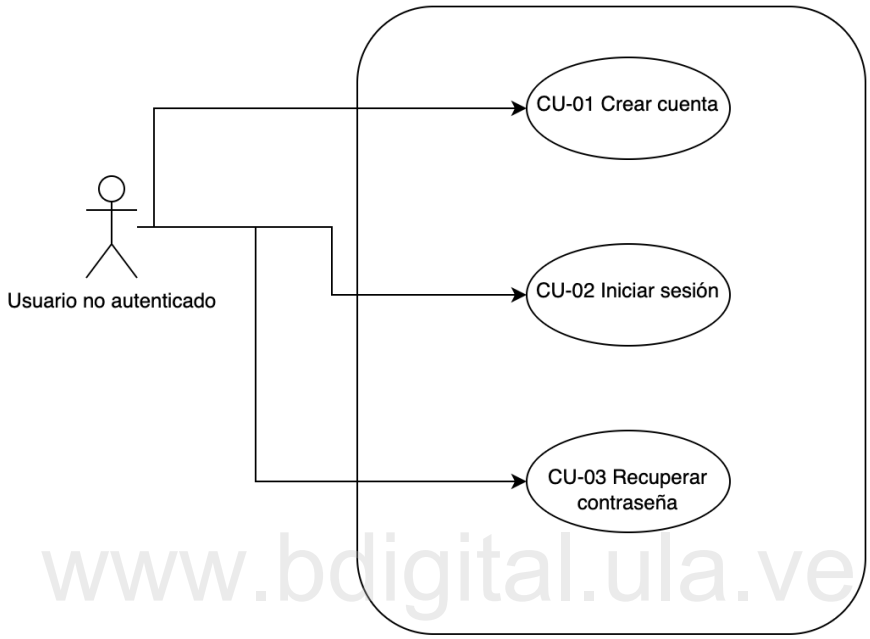


Figura 3.2: Diagrama de Casos de Uso 01 - Iniciar sesión, crear cuenta y recuperar contraseña.

Caso de Uso - Iniciar sesión, crear cuenta y recuperar contraseña: en las tablas 3.5,3.6 y 3.7 contienen la descripción textual detallada de los casos de uso mostrados en la Figura3.2.

Tabla 3.5: Descripción del Caso de Uso Crear Cuenta.

Nombre	CU - 01 Crear cuenta.
Descripción	Este caso de uso ocurre cuando un usuario no autenticado desea registrarse en la aplicación.
Actor	Usuario no Autenticado.
Pre-condiciones	Ninguna.

Continua en la siguiente página.

Tabla 3.5 – *Continuación de la página anterior.*

Flujo normal	1. El actor solicita al sistema que quiere Registrarse. 2. El sistema muestra la pantalla de selección de rol y solicita al usuario que seleccione uno bajo el cual se quiere registrar (este puede ser administrador,propietario o vigilante). 3. El actor selecciona uno de los tres roles. 4. El sistema muestra la pantalla de registro y solicita al usuario los datos. 5. El actor ingresa los datos de registro. 6. El sistema verifica la validez de los datos de registro y los almacena. 7. El sistema muestra un mensaje al usuario indicando que ha sido registrado.
Flujo alternativo	6.1 Si los datos de registro tienen un formato inválido, el sistema lo notifica y regresa al paso 4.
Post-condiciones	El usuario ha sido registrado en el sistema bajo un rol y puede iniciar sesión con su cuenta.

Tabla 3.6: Descripción del Caso de Uso Iniciar sesión.

Nombre	CU - 02 Iniciar sesión.
Descripción	Este caso de uso ocurre cuando un usuario no autenticado desea iniciar sesión en la aplicación.
Actor	Usuario no Autenticado.
Pre-condiciones	Ninguna.
Flujo normal	1. El actor solicita al sistema que quiere Iniciar sesión. 2. El sistema muestra la pantalla de inicio sesión y solicita al usuario que introduzca los datos de inicio de sesión (correo electrónico y contraseña). 3. El actor introduce los datos de inicio de sesión. 4. El sistema verifica la validez de los datos de inicio de sesión. 5. El sistema muestra un mensaje al usuario indicándole que ha iniciado sesión correctamente. 6. El sistema muestra la pantalla principal.

Continua en la siguiente página.

Tabla 3.6 – *Continuación de la página anterior.*

Flujo alternativo	**4.**1. Si los datos de inicio de sesión tienen un formato inválido, el sistema lo notifica y regresa al paso 2. 4.2. Si los datos de inicio de sesión tienen un formato inválido, el sistema lo notifica y regresa al paso 2.
Post-condiciones	El usuario ha sido registrado en el sistema bajo un rol y puede iniciar sesión con su cuenta.

Tabla 3.7: Descripción del Caso de Uso Recuperar Contraseña.

Nombre	CU - 03 Recuperar Contraseña.
Descripción	Este caso de uso ocurre cuando un usuario no recuerda su contraseña y deseea recuperar su acceso a la aplicación.
Actor	Usuario no Autenticado.
Pre-condiciones	Ninguna.
Flujo normal	1. El actor solicita al sistema que quiere recuperar la contraseña. 2. El sistema muestra la pantalla de recuperar la contraseña y solicita al usuario que introduzca los datos de recuperar la contraseña (correo electrónico). 3. El actor introduce su correo electrónico. 4. El sistema verifica la validez del correo electrónico y envía el código de recuperación de contraseña. 5. El actor ingresa el código de recuperación correctamente. 6. El sistema verifica la validez del código de recuperación ingresado. 7. El sistema solicita al usuario que ingrese la nueva contraseña y su confirmación. 8. El sistema verifica la nueva contraseña y la almacena. 9. El sistema notifica al usuario que el restablecimiento de contraseña ha sido exitoso.

Continua en la siguiente página.

Tabla 3.7 – *Continuación de la página anterior.*

Flujo alternativo	**4.**1. Si el correo electrónico tiene un formato inválido, el sistema lo notifica y regresa al paso 2. 4.2. Si el correo electrónico no pertenece a un usuario registrado, el sistema lo notifica y regresa al paso 2. **7.**1. Si el código de recuperación tiene un formato inválido, el sistema le notifica al actor y regresa al paso 5. 7.2. Si el código de recuperación ha expirado el sistema lo notifica y regresa al paso 2. **9.**1. Si la contraseña tienen un formato inválido o no coinciden, el sistema lo notifica y regresa al paso 8.
Post-condiciones	El usuario ha sido registrado en el sistema bajo un rol y puede iniciar sesión con su cuenta.

Diagramas de Actividades, crear cuenta y recuperar contraseña: en las figuras 3.3,3.4 y 3.5 muestran los diagramas de actividades para los casos de usos descriptos anteoriormente. mostrados en la Figura3.2.

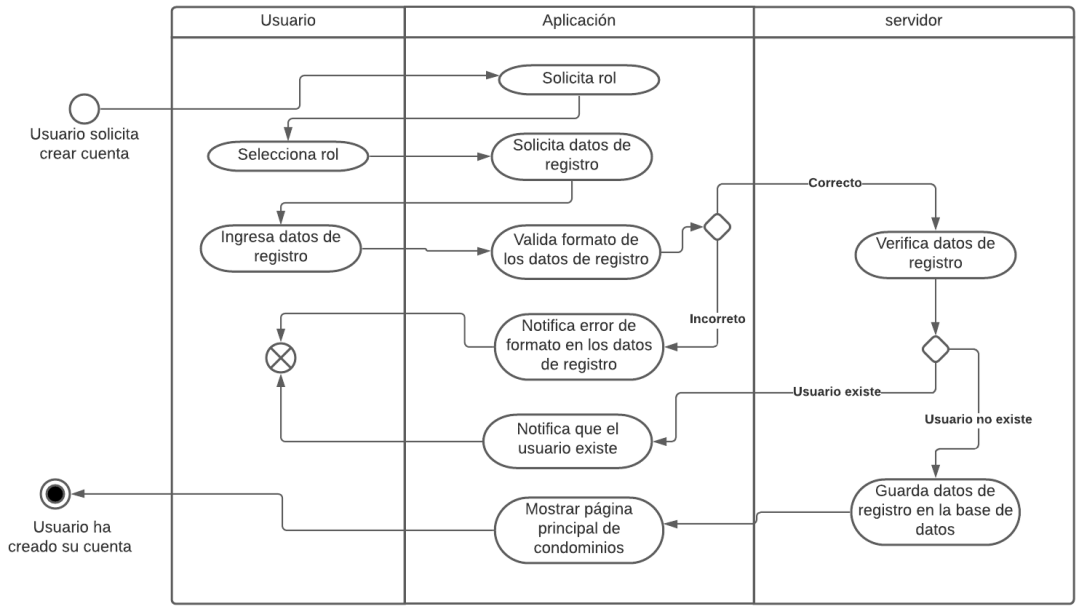


Figura 3.3: Diagrama de Actividades Caso de uso CU-01. Crear cuenta

Diagramas de Secuencias, crear cuenta y recuperar contraseña: en las figuras 3.6,3.7 y 3.8 muestran los diagramas de secuencia para los casos de usos descriptos anteoriormente. mostrados en la Figura3.2.

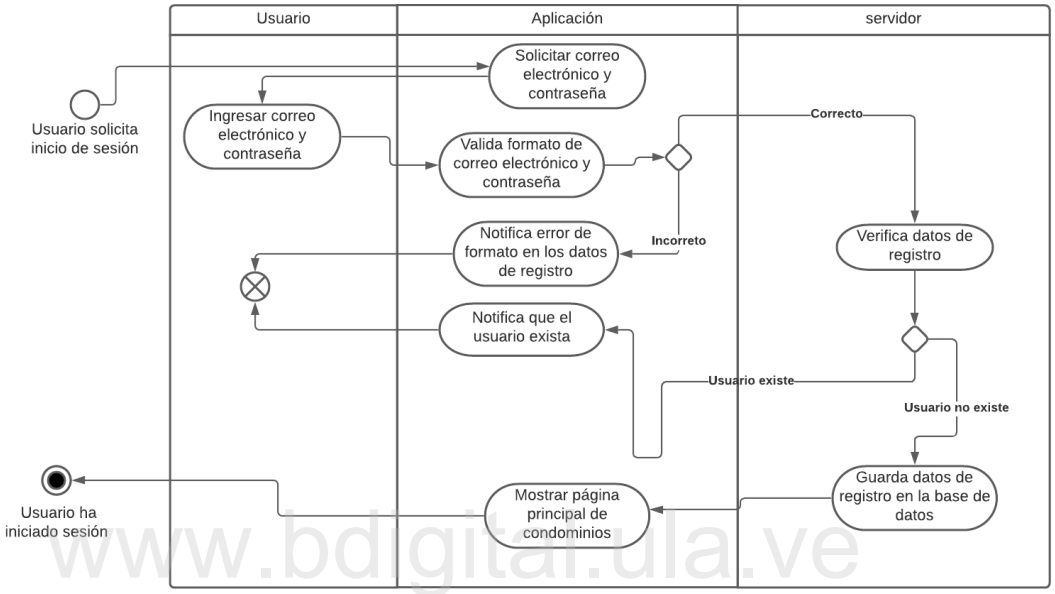


Figura 3.4: Diagrama de Actividades Caso de uso CU-02. Iniciar sesión

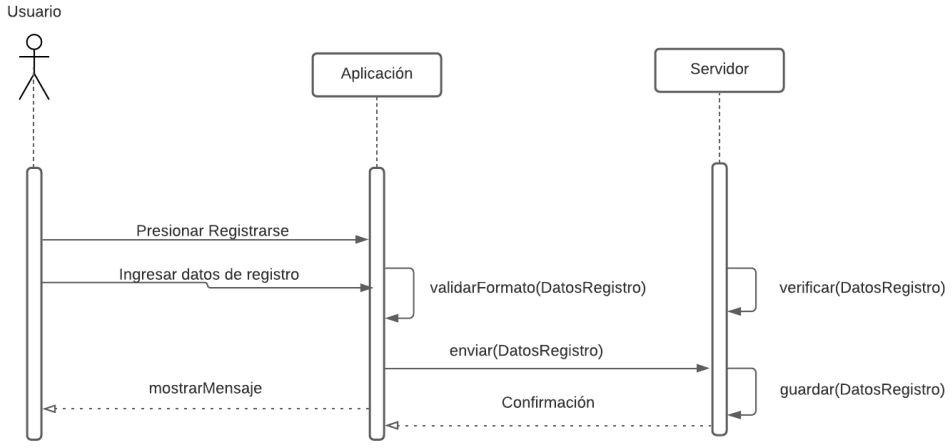


Figura 3.6: Diagrama de Secuencia Caso de uso CU-01. Crear cuenta

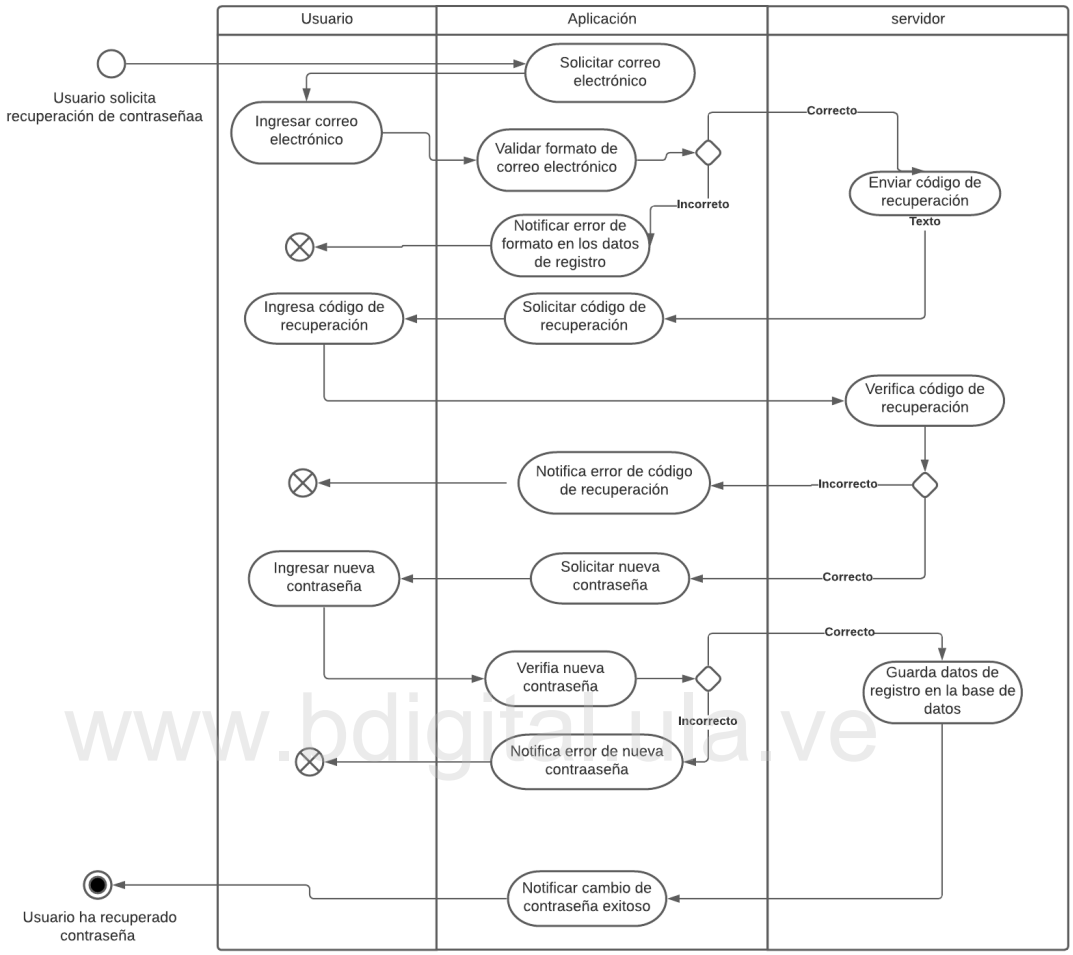


Figura 3.5: Diagrama de Actividades Caso de uso CU-03. Recuperar contraseña

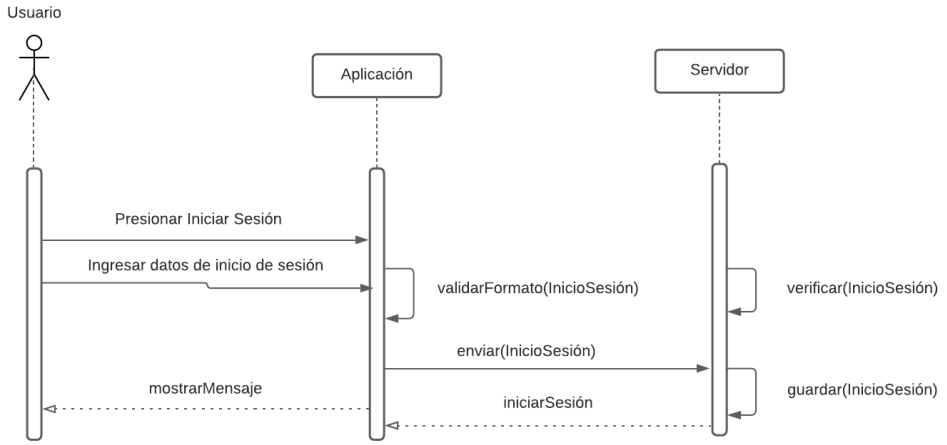


Figura 3.7: Diagrama de Secuencia Caso de uso CU-02. Iniciar sesión

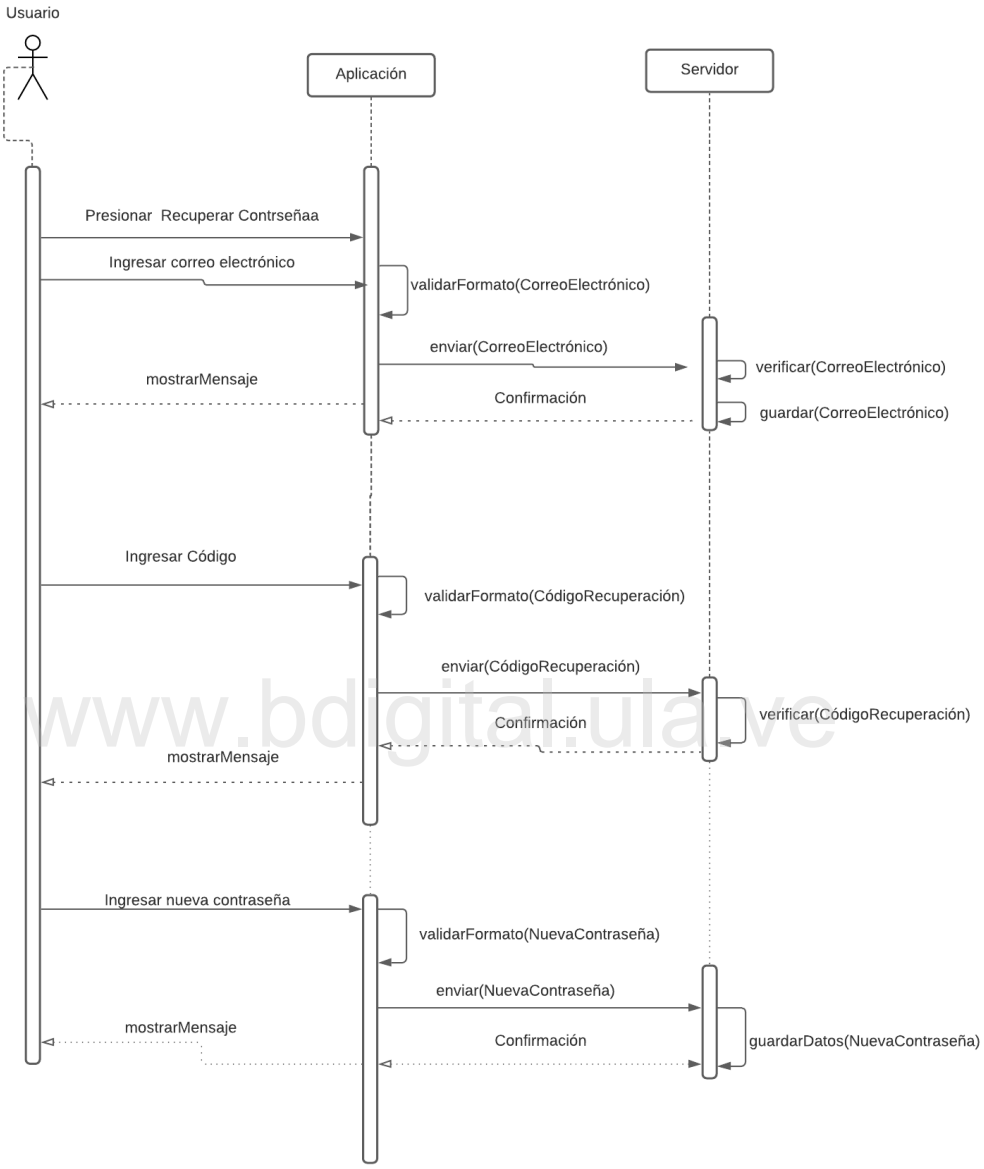


Figura 3.8: Diagrama de Secuencia Caso de uso CU-03. Recuperar contraseña

3.2.2.2. Gestionar cuenta:

La gestión de cuenta perfil tendrá tres operaciones fundamentales; entre estas operaciones están editar perfil (donde podrá observar la información del perfil almacenada en la base datos), cambiar contraseña y cerrar sesión.

Cualquier usuario atenticado como Administrador, propierario o vigilante podrá acceder a la gestión de cuenta. En la Figura3.9 se muestra el diagrama de caso de uso para las operaciones descritas anteriormente.

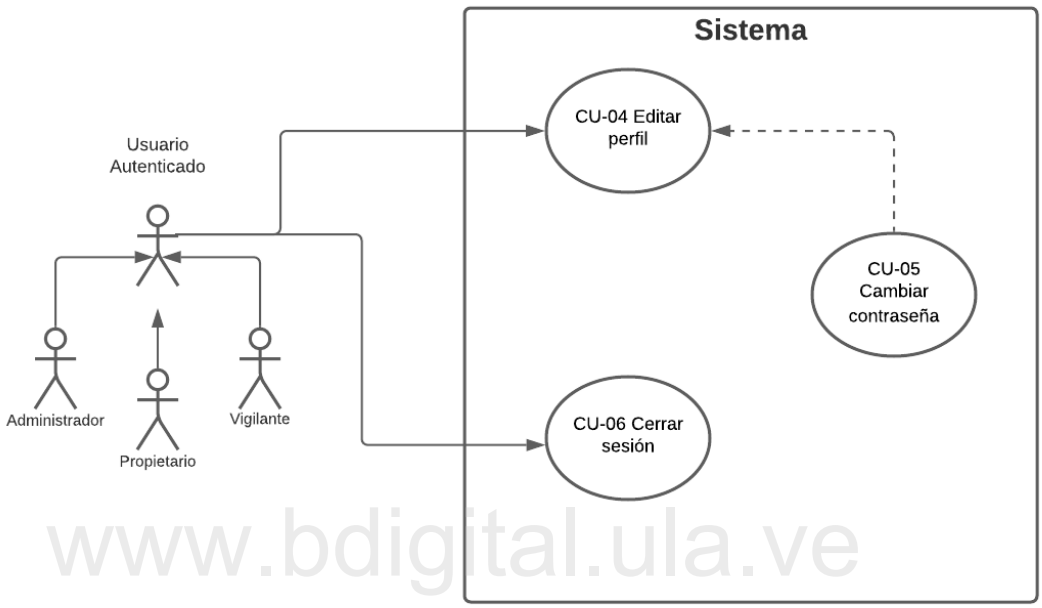


Figura 3.9: Diagrama de Caso de uso CU-03. Administrar perfil

Caso de Uso - Gestionar cuenta: en las tablas 3.8,3.9 y 3.10 contienen la descripción textual detallada de los casos de uso mostrados en la Figura3.9.

Tabla 3.8: Descripción del Caso de Uso Editar perfil.

Nombre	CU - 04 Editar perfil.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario autenticado puede editar información de perfil.
Actor	Administrador, Propietario, Vigilante.
Pre-condiciones	Ninguna.

Continua en la siguiente página.

Tabla 3.8 – *Continuación de la página anterior.*

Flujo normal	1. El actor solicita al sistema que quiere editar su peril. 2. El sistema solicita los datos actualizados. 3. El actor ingresa los datos de actualización. 4. El sistema valida los datos de actualización. 5. El sistema guarda los datos de actuallización en la base de datos. 6. El sistema muestra mensaje de que se actualizaron correctamente.
Flujo alternativo	1. Si el actor introduce datos de actualización erroneos, el sistema notifica y vuelve al paso 2.
Post-condiciones	El usuario ha editado su información de perfil.

Tabla 3.9: Descripción del Caso de Uso Cambiar contraseña.

Nombre	CU - 05 Cambiar contraseña.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario autenticado puede cambiar la contraseña.
Actor	Administrador, Propietario, Vigilante.
Pre-condiciones	CU - 04 Editar perfil.
Flujo normal	1. El actor solicita al sistema que quiere editar contraseña. 2. El sistema solicita la contraseña actual. 3. El actor ingresa la contraseña actual, la nueva contraseña y la confirmación. 4. El sistema verifica la contraseña actual, la nueva contraseña y la confirmación y le informa al usuario que la operación fue exitosa.

Continua en la siguiente página.

Tabla 3.9 – *Continuación de la página anterior.*

Flujo alternativo	1. Si el actor introduce una contraseña actual incorrecta, el sistema lo notifica y vuelve al paso 2. 1. Si la nueva contraseña y confirmación no coinciden, el sistema le notifica al usuario y le permite al usuario corregirlo.
Post-condiciones	Ninguno.

Tabla 3.10: Descripción del Caso de Uso Cerrar sesión.

Nombre	CU - 06 Cerrar sesión.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario autenticado cierra su sesión.
Actor	Administrador, Propietario, Vigilante.
Pre-condiciones	CU - 04 Editar perfil.
Flujo normal	1. El actor solicita al sistema que quiere cerrar sesión. 2. El sistema muestra un mensaje al usuario para que confirme si realmente desea cerrar sesión. 3. El actor confirma la operaación. 4. El sistema notifica al actor que ha cerrado sesión correctamente y muestra la pantalla de inicio.
Flujo alternativo	1. Si el actor cancela la operación la sesión no se cierra y este caso de uso termina.
Post-condiciones	El actor ha cerrado sesión, por lo que se convierte en un usuario no autenticado.

Diagramas de Actividades - Gestionar cuenta: en las figuras 3.10, 3.11 y 3.12 muestran los diagramas de los casos de uso CU-04, CU-05 y CU.06

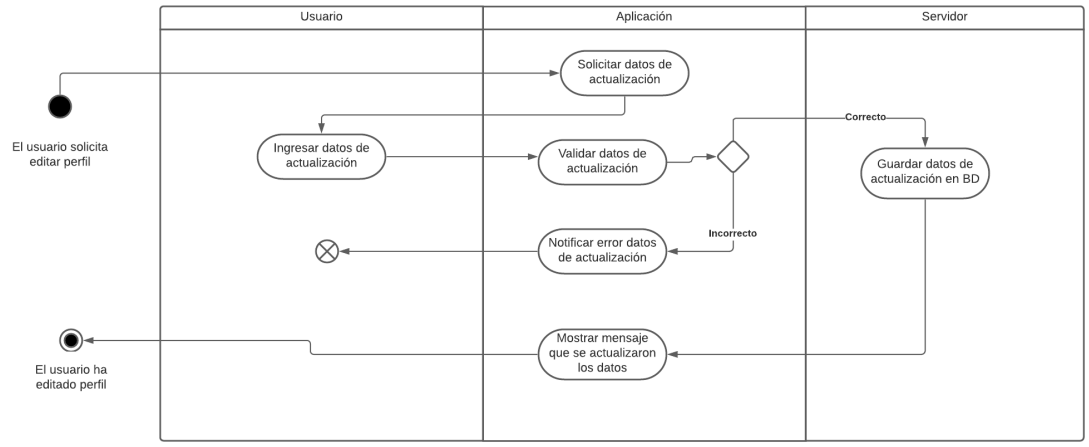


Figura 3.10: Diagrama de Actividades Caso de uso CU-04. Editar perfil

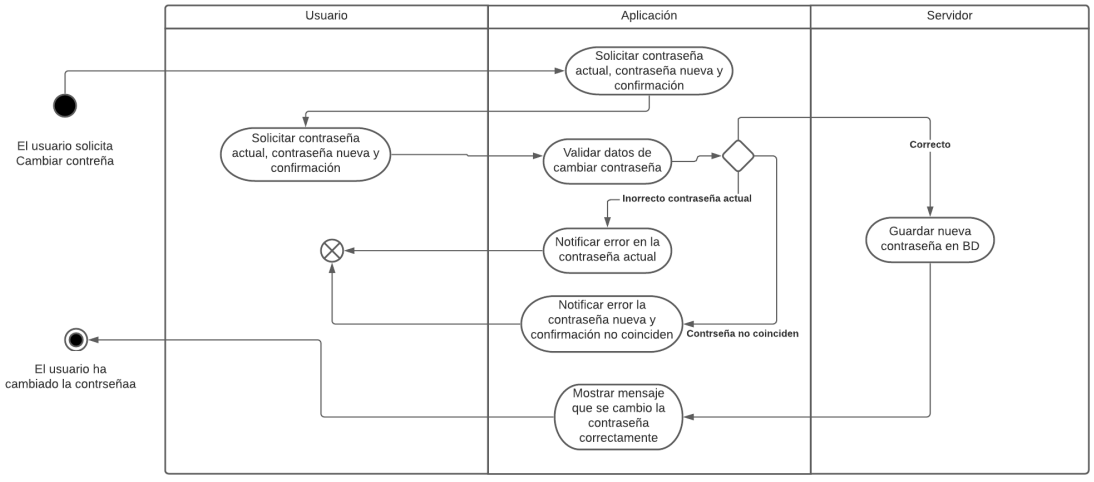


Figura 3.11: Diagrama de Actividades Caso de uso CU-05. Cambiar contraseña

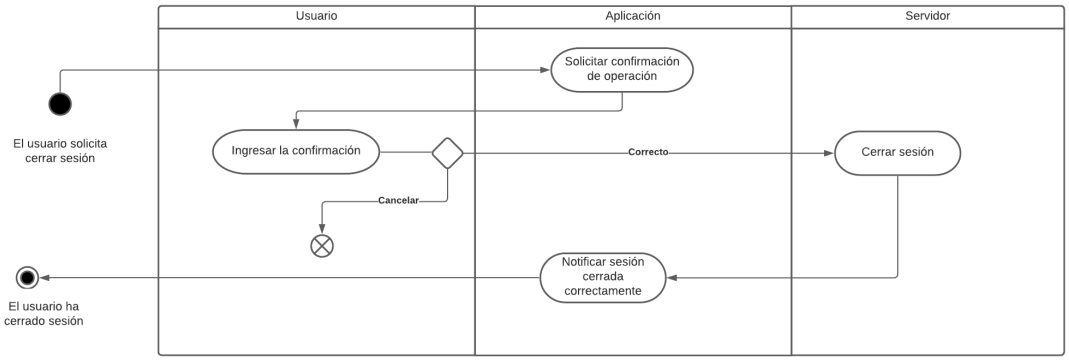


Figura 3.12: Diagrama de Actividades Caso de uso CU-06. Cerrar sesión

Diagramas Secuencia - Gestionar cuenta: en las figuras 3.13, 3.14 muestran los diagramas de secuencia de los casos CU-04, CU-05 y CU.06

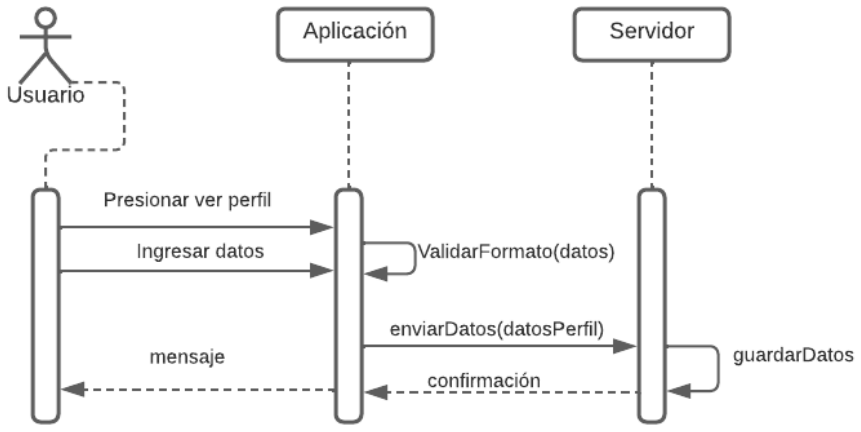


Figura 3.13: Diagrama de Secuencia Caso de uso CU-04. Editar perfil

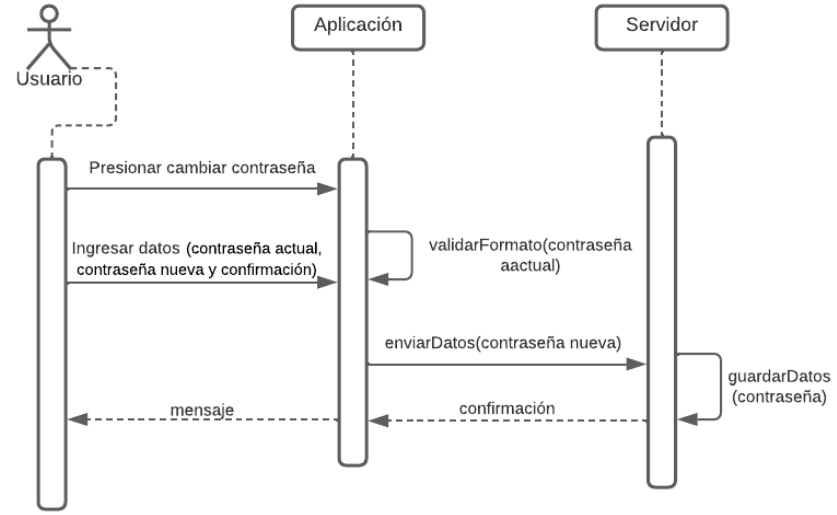


Figura 3.14: Diagrama de Secuencia Caso de uso CU-05. Cambiar contraseña

3.2.2.3. Gestión de condominios:

Los usuarios que se registran como administradores tienen acceso a la gestión de condominios, pueden crear, buscar, editar o eliminar condominios. En la figura 3.15 se muestra el diagrama de caso de uso correspondiente a estas operaciones

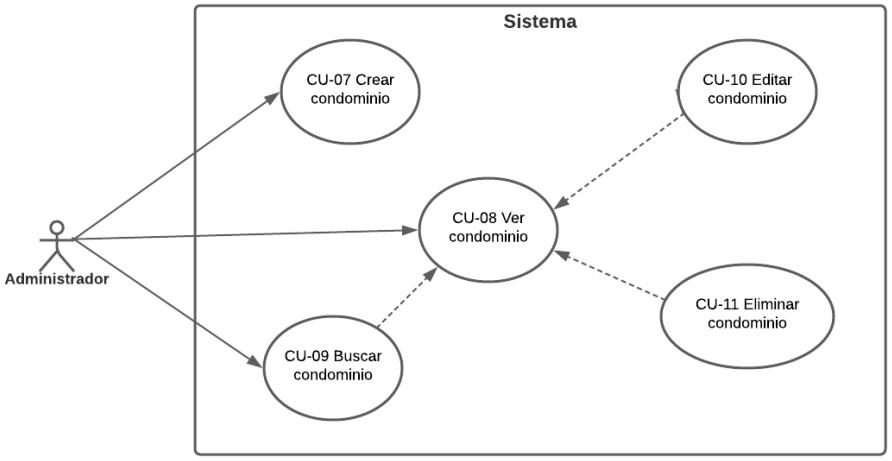


Figura 3.15: Diagrama de Caso de uso CU-03 Gestión de condominios

Caso de Uso - Gestión de condominios: en las tablas 3.11, 3.12,3.13,3.14 y 3.15 contienen la descripción textual detallada de los casos de uso mostrados en la Figura3.15.

Tabla 3.11: Descripción del Caso de Uso Crear condominio.

Nombre	CU - 07 Crear condominio.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita crear un perfil.
Actor	Administrador.
Pre-condiciones	Ninguna.
Flujo normal	1. El actor solicita al sistema que quiere crear un condominio. 2. El sistema solicita los datos del condiminio. 3. El actor ingresa los datos del condiminio. 4. El sistema valida los datos del condiminio. 5. El sistema guarda los datos del condiminio en base de datos. 6. El sistema muestra mensaje que se creó correctamente.
Flujo alternativo	1. Si el actor introduce datos del condminio incorrectos, el sistema lo notifica y vuelve al paso 2.
Post-condiciones	El usuario ha creado un condominio.

Tabla 3.12: Descripción del Caso de Uso Buscar condominio.

Nombre	CU - 08 Buscar condominio.
Descripción	Este caso de uso ocurre cuando un usuario realiza una busqueda de condominios, para ello debe ingresar parámetros de búsqueda como nombre o código de condominio.
Actor	Administrador.
Pre-condiciones	Ninguna.

Continua en la siguiente página.

Tabla 3.12 – *Continuación de la página anterior.*

Flujo normal	1. El actor solicita al sistema que quiere buscar un condominio. 2. El sistema solicita muestra la pantalla de búsqueda y solicita que introduzca los parámateros de búsqueda. 3. El actor los parámateros de búsqueda. 4. El sistema váalida los parámateros de búsqueda y realiza una búsqueda en la base de datos. 5. El sistema muestra la pantalla con los condominios que coincidan con los parámateros de búsqueda.
Flujo alternativo	1. Si los parámateros de búsqueda no tiene un formato válido lo notifica y vuelve al paso 2. 2. Sino se consiguen ningún condominio que coincida con los parámateros de búsqueda, el sistema lo notifica y se termina el caso de uso.
Post-condiciones	Ninguna.

Tabla 3.13: Descripción del Caso de Uso Ver condominio.

Nombre	CU - 09 Ver condominio.
Descripción	Este caso de uso ocurre cuando un usuario solicita al sistema ver un condiminio.
Actor	Administrador.
Pre-condiciones	Ninguna.
Flujo normal	1. El actor solicita al sistema que quiere ver un condominio. 2. El sistema muestra una pantalla con la información del condiminio almacenada en la base de datos.
Flujo alternativo	Ninguno.
Post-condiciones	Ninguna.

Tabla 3.14: Descripción del Caso de Uso Editar condominio.

Nombre	CU - 10 Editar condominio.
Descripción	Este caso de uso ocurre cuando un usuario solicita al sistema editar un condiminio.
Actor	Administrador.
Pre-condiciones	CU-09 Ver Condominio.
Flujo normal	1. El actor solicita al sistema que quiere editar un condominio. 2. El sistema solicita los datos del condominio actuallizados. 3. El actor ingresa los datos del condominio actuallizados. 4. El sistema válida los datos del condominio actuallizados y los guarda los datos en base de datos. 5. El sistema informa que los datos del condominio han sido actuallizados.
Flujo alternativo	Ninguno.
Post-condiciones	El usuario ha editado información de un condominio.

Tabla 3.15: Descripción del Caso de Uso Eliminar condominio.

Nombre	CU - 11 Eliminar condominio.
Descripción	Este caso de uso ocurre cuando un usuario solicita al sistema eliminar un condiminio.
Actor	Administrador.
Pre-condiciones	CU-09 Ver Condominio.
Flujo normal	1. El actor solicita al sistema que quiere eliminar un condominio. 2. El sistema solicita confirmación para eliminar el condominio. 3. El actor confirma la solicitud. 4. El sistema elimina el condominio y notifica al usuario que se elimino correctamente.
Flujo alternativo	El usuario ha eliminado el condominio.

Continua en la siguiente página.

Tabla 3.15 – *Continuación de la página anterior.*

Post-condiciones	El usuario ha editado información de un condominio.

Diagramas de Actividades - Gestión de condominios: en las figuras 3.16, 3.173.18,3.19 y 3.20 muestran los diagramas de los casos de uso CU-07, CU-08, CU-09, CU-10 y CU-11

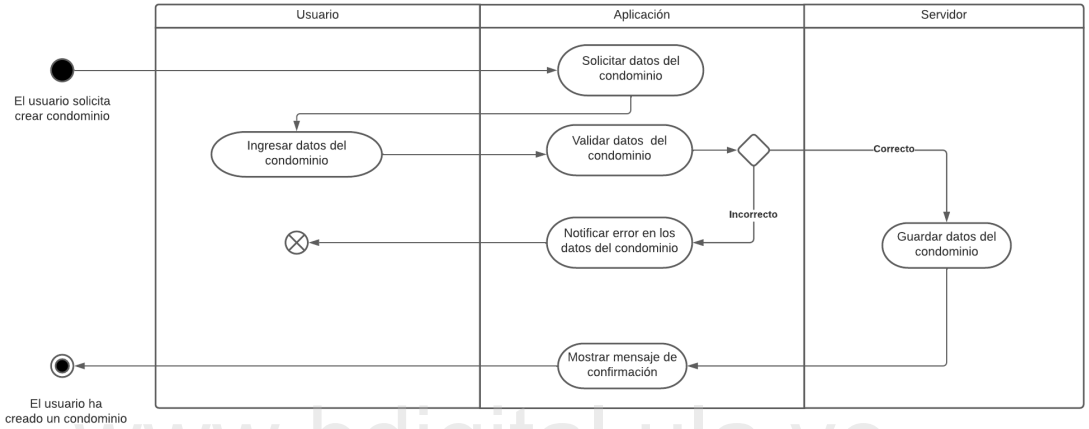


Figura 3.16: Diagrama de Actividad Caso de uso CU-07. Crear Condominio

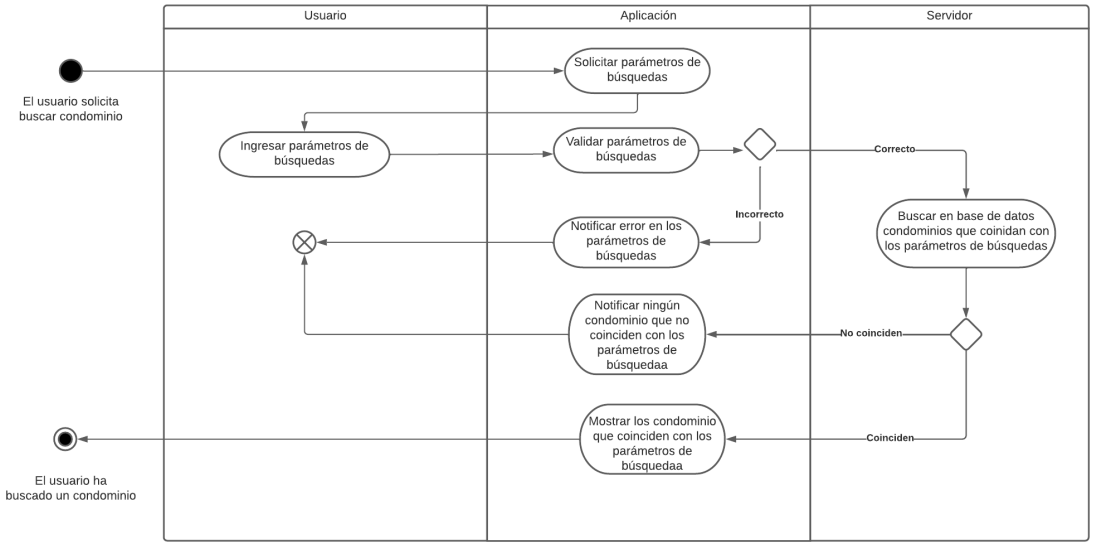


Figura 3.17: Diagrama de Actividad Caso de uso CU-08. Buscar Condominio

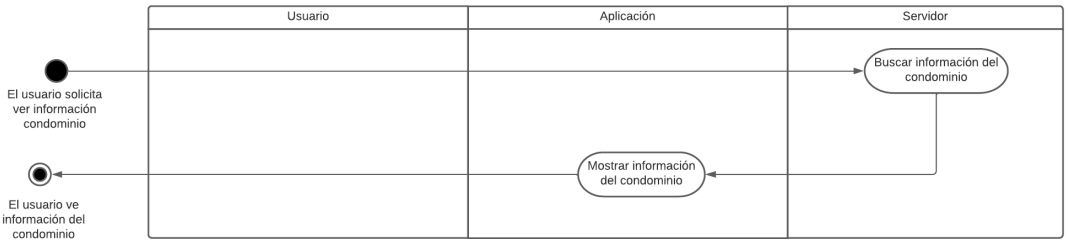


Figura 3.18: Diagrama de Actividad Caso de uso CU-09. Ver Condominio

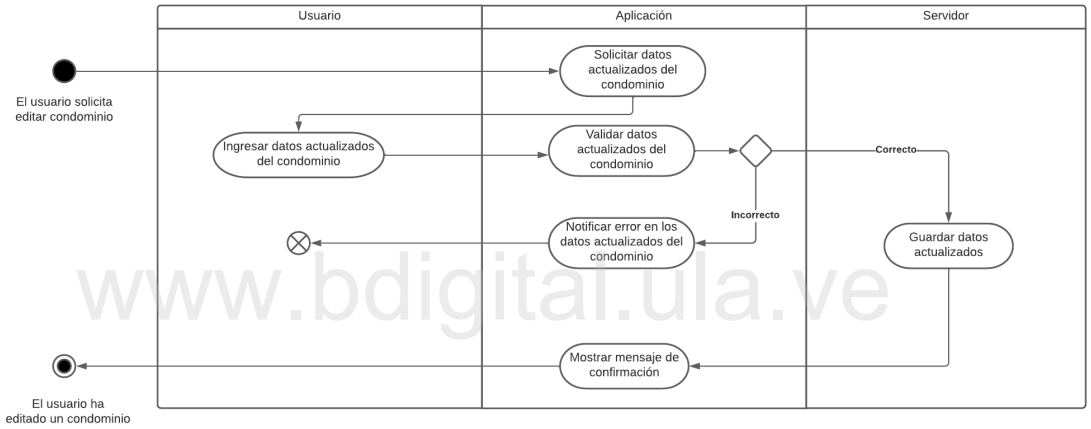


Figura 3.19: Diagrama de Actividad Caso de uso CU-10. Editar Condominio

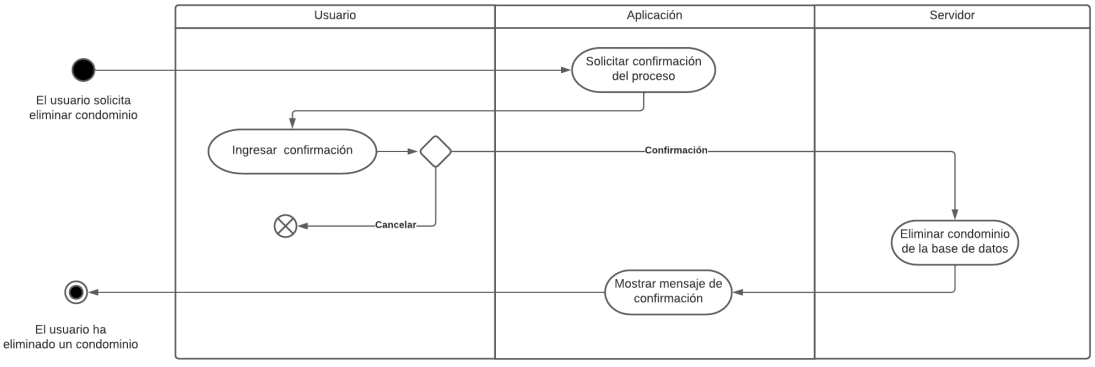


Figura 3.20: Diagrama de Actividad Caso de uso CU-11. Eliminar Condominio

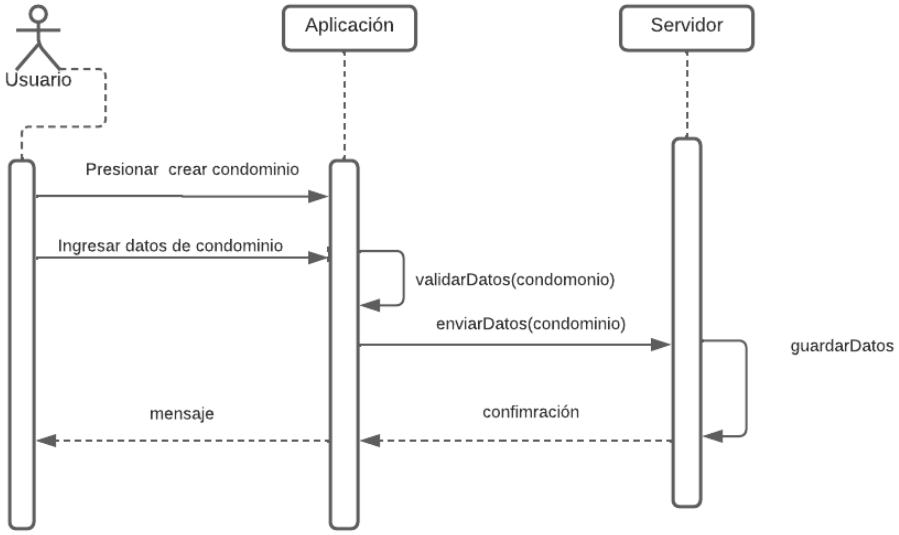


Figura 3.21: Diagrama de Secuencia Caso de uso CU-07. Crear Condominio

Diagramas Secuencia - Gestión de condominios: en las figuras 3.21, 3.22,3.23,3.24 y 3.25 muestran los diagramas de secuencia de los casos CU-07, CU-08, CU-09, CU-10 y CU-11.

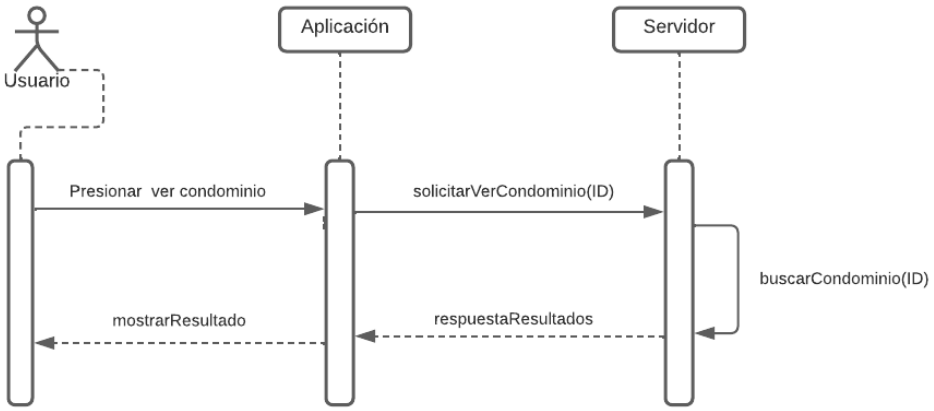


Figura 3.23: Diagrama de Secuencia Caso de uso CU-09. Ver Condominio

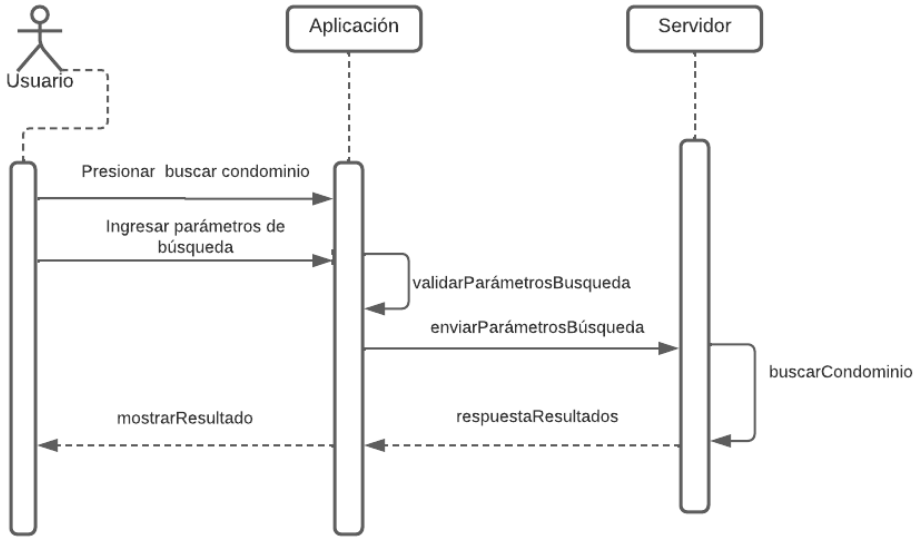


Figura 3.22: Diagrama de Secuencia Caso de uso CU-08. Buscar Condominio

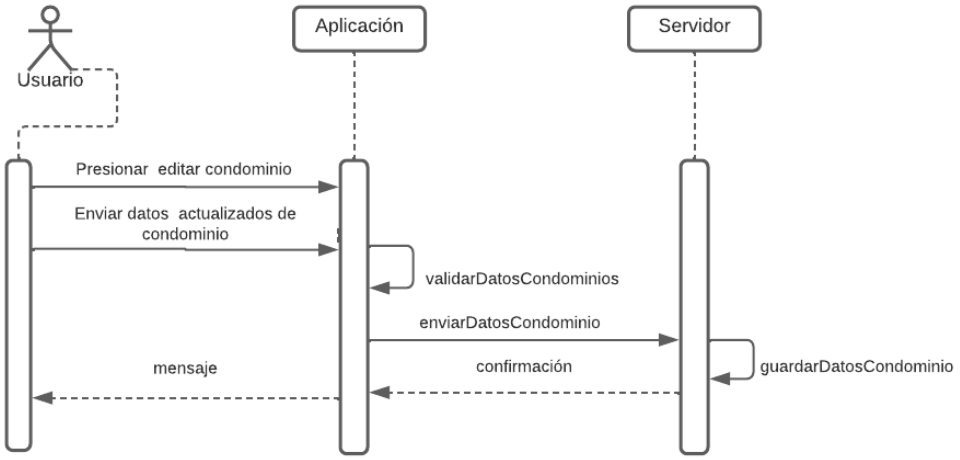


Figura 3.24: Diagrama de Secuencia Caso de uso CU-10. Editar Condominio

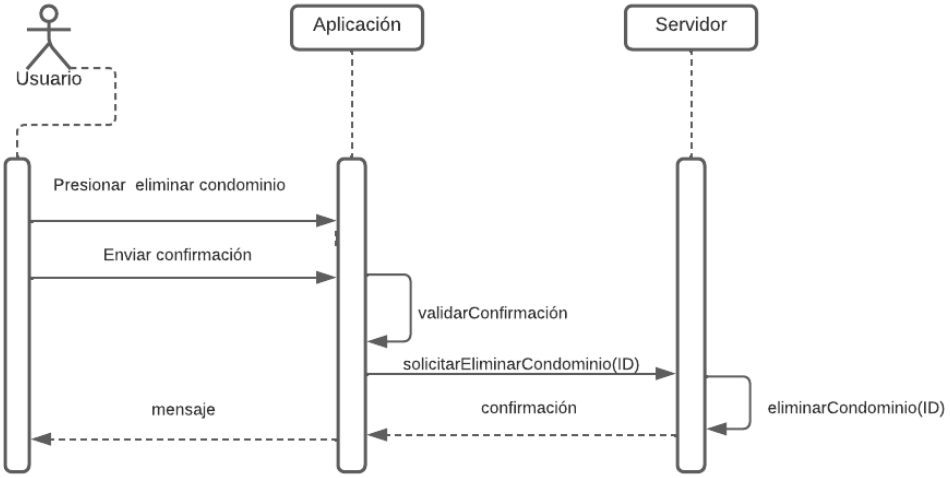


Figura 3.25: Diagrama de Caso de uso CU-11. Eliminar Condominio

3.2.2.4. Gestión de ingresos/egresos:

Aquí se incluyen las operaciones que permiten a los administradores gestionar los ingresos/egresos que tiene un condominio; esto incluye operaciones de crear nuevos, editar, eliminar o listar ingresos/egresos. Es importante resaltar el administrador puede listar los ingresos/egresos por parámteros de búsqueda.

Caso de Uso - Gestión de ingresos/egresos: en las tablas 3.16, 3.17,3.18 y 3.19 contienen la descripción textual detallada de los casos de uso mostrados en la Figura3.26.

Tabla 3.16: Descripción del Caso de Uso Crear Ingresos/Egresos.

Nombre	CU - 12 Crear Ingresos/Egresos.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita crear un Ingresos/Egresos.
Actor	Administrador.
Pre-condiciones	Ninguna.

Continua en la siguiente página.

Tabla 3.16 – *Continuación de la página anterior.*

Flujo normal	1. El actor solicita al sistema que quiere crear un ingreso/Egreso. 2. El sistema solicita los datos del ingreso/egreso. 3. El actor ingresa los datos del ingreso/egreso. 4. El sistema valida los datos del ingreso/egreso. 5. El sistema guarda los datos del ingreso/egreso en base de datos. 6. El sistema muestra mensaje que se creó correctamente.
Flujo alternativo	3.1 Si el actor introduce datos del ingreso/egreso incorrectos, el sistema lo notifica y vuelve al paso 2.
Post-condiciones	El usuario ha creado un ingreso/egreso.

Tabla 3.17: Descripción del Caso de Uso Listar Ingresos/Egresos.

Nombre	CU - 13 Listar Ingresos/Egresos.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita listar los Ingresos/Egresos.
Actor	Administrador.
Pre-condiciones	Ninguna.
Flujo normal	1. El actor solicita al sistema listar los ingresos/Egresos. 2. El sistema solicita los parámetros de búsqueda. 3. El actor ingresa los parámetros de búsqueda. 4. El sistema valida los parámetros de búsqueda. 5. El sistema realiza la búsqueda de ingresos/egresos que coincidan con los parámetros de búsqueda. 6. El sistema muestra el listado de ingresos/egresos.

Continua en la siguiente página.

Tabla 3.17 – *Continuación de la página anterior.*

Flujo alternativo	3.1 Si el actor introduce parámetros de búsqueda incorrectos, el sistema lo notifica y vuelve al paso 2. 5.1 Si el sistema no consigue ingresos/egresos que coincidan con los parámetros de búsqueda, lo notifica y termina el caso de uso .
Post-condiciones	Ninguna.

Tabla 3.18: Descripción del Caso de Uso Editar Ingreso/Egreso.

Nombre	CU - 14 Editar Ingreso/Egreso.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita editar un ingreso/egreso.
Actor	Administrador.
Pre-condiciones	CU - 13 Listar Ingresos/Egresos.
Flujo normal	1. El actor solicita al sistema editar un ingreso/egreso. 2. El sistema solicita los datos actualizados. 3. El actor ingresa los datos actualizados. 4. El sistema válida los datos actualizados. 5. El sistema guarda los datos actualizados en la base de datos y notifica que se guardaron correctamente.
Flujo alternativo	3.1 Si el actor introduce datos actualizados inválidos, el sistema lo notifica y vuelve al paso 2.
Post-condiciones	Ninguna.

Tabla 3.19: Descripción del Caso de Uso Eliminar Ingreso/Egreso.

Nombre	CU - 15 Eliminar Ingreso/Egreso.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita elliminar un ingreso/egreso.
Actor	Administrador.
Pre-condiciones	CU - 13 Listar Ingresos/Egresos.

Continua en la siguiente página.

Tabla 3.19 – *Continuación de la página anterior.*

Flujo normal	1. El actor solicita al sistema eliminar un ingreso/egreso. 2. El sistema solicita confirmación de la solicitud. 3. El actor confirma la solicitud. 4. El sistema elimina el ingreso/egreso.
Flujo alternativo	3.1 Si el actor cancela la confirmación el caso de uso termina.
Post-condiciones	Ninguna.

Diagramas de Actividades - Gestión de ingresos/egresos: en las figuras 3.27, **??**3.29,3.30 muestran los diagramas de los casos de uso CU-12, CU-13, CU-14 y CU-15

Diagramas Secuencia - Gestión de ingresos/egresos: en las figuras 3.21, 3.31,3.32,3.33 y 3.34 muestran los diagramas de secuencia de los casos CU-12, CU-13, CU-14 y CU-15.

3.2.2.5. Gestión de pagos pendientes:

Los propietario pueden listar los pagos pendientes respectivos a su usuario además solicitar la confirmación de los mismos, los administradores podrán confirmar la solicitud y cambiar el estado de pago a recibido.

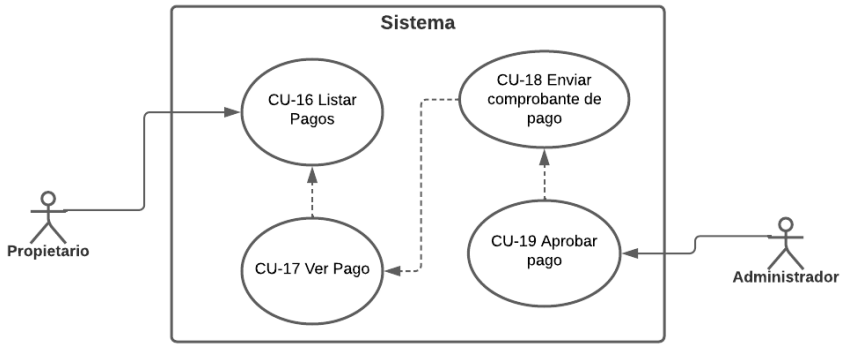


Figura 3.35: Diagrama de Caso de uso CU-05 Gestión de pagos

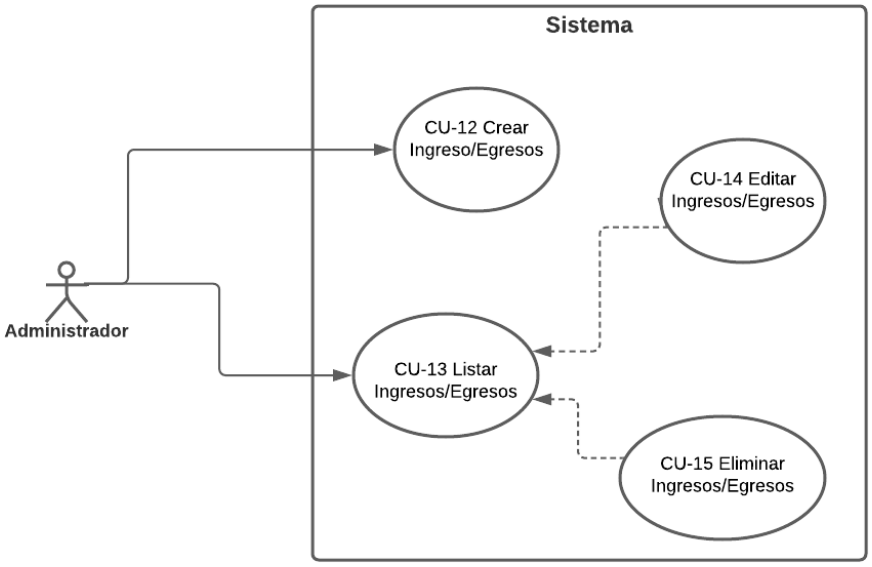


Figura 3.26: Diagrama de Caso de uso CU-12 Gestión de ingresos/egresos

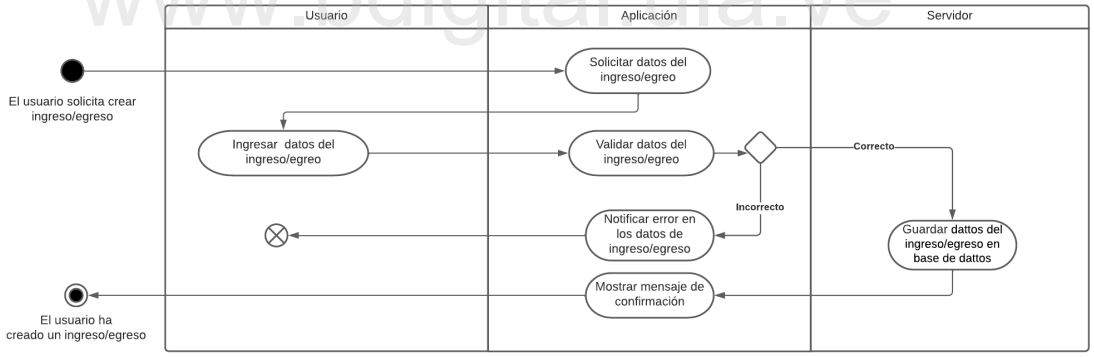


Figura 3.27: Diagrama de Actividad Caso de uso CU-12. Crear Ingreso/Egreso

Caso de Uso - Gestión de pagos pendientes: en las tablas 3.20, 3.21,3.22 y 3.23 contienen la descripción textual detallada de los casos de uso mostrados en la Figura3.35.

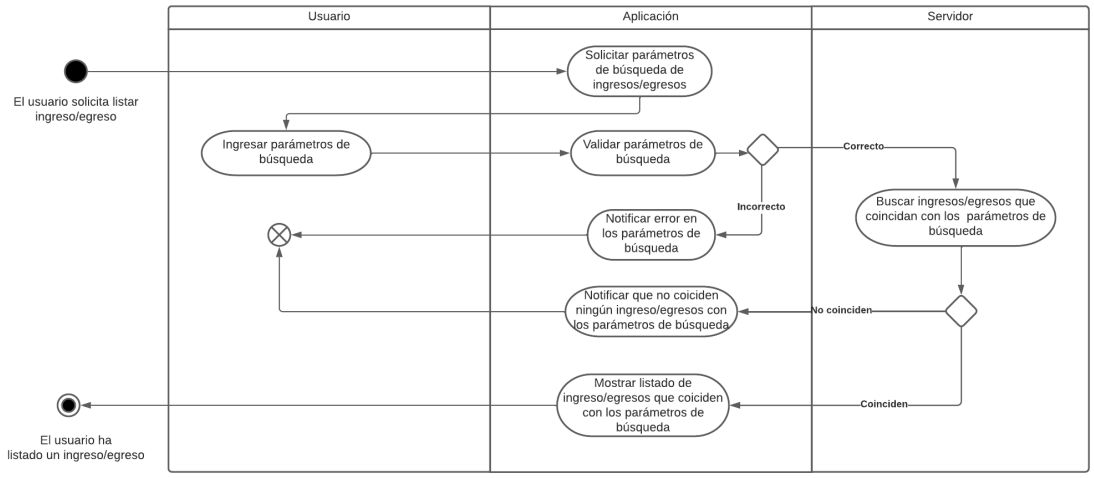


Figura 3.28: Diagrama de Actividad Caso de uso CU-13. Listar Ingreso/Egreso

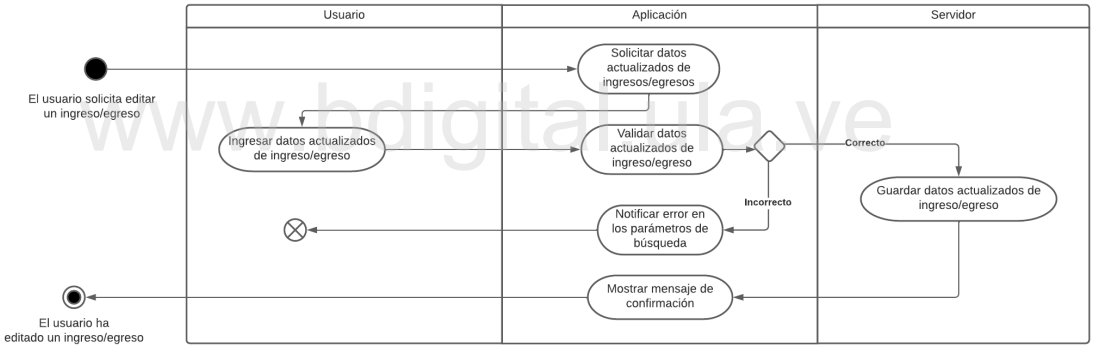


Figura 3.29: Diagrama de Actividad Caso de uso CU-14. Editar Ingreso/Egreso

Tabla 3.20: Descripción del Caso de Uso Listar Pagos Pendientes.

Nombre	CU - 16 Listar Pagos.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita al sistema listar los pagos pendientes.
Actor	Propietaario.
Pre-condiciones	Ninguna.

Continua en la siguiente página.

Tabla 3.20 – *Continuación de la página anterior.*

Flujo normal	1. El actor solicita al sistema que quiere listar los pagos. 2. El sistema busca en base de datos los pagos pendientes relacionados con el actor. 3. El sistema muestra una pantalla con el listado de pagos.
Flujo alternativo	Ninguna.
Post-condiciones	Ninguna.

Tabla 3.21: Descripción del Caso de Uso Ver Pagos.

Nombre	CU - 17 Ver Pagos.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita al sistema ver los pagos.
Actor	Propietaario.
Pre-condiciones	CU - 16 Listar Pagos.
Flujo normal	1. El actor solicita al sistema que quiere ver un pago. 2. El sistema busca en base de datos la información relacionada con el pago. 3. El sistema muestra una pantalla con la información del pago.
Flujo alternativo	Ninguna.
Post-condiciones	Ninguna.

Tabla 3.22: Descripción del Caso de Uso Enviar comprobante de pago.

Nombre	CU - 18 Enviar comprobante de pago.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita al sistema enviar comprobante de pago.
Actor	Propietaario.
Pre-condiciones	CU - 17 Ver Pagos.

Continua en la siguiente página.

Tabla 3.22 – *Continuación de la página anterior.*

Flujo normal	1. El actor solicita al sistema que quiere enviar comprobante de pago. 2. El sistema solicita los datos del comprobante. 3. El actor ingresa los datos del comprobante. 4. El sistema válida los datos del comprobante. 5. El sistema cambia el estado a revisión del pago y notifica que se recibieron los datos.
Flujo alternativo	4.1 Si el actor introduce datos del comprobante incorrectos, el sistema lo notifica y vuelve al paso 2.
Post-condiciones	El usuario ha enviado el comprobante de pago.

Tabla 3.23: Descripción del Caso de Uso Enviar comprobante de pago.

Nombre	CU - 19 Aprobar pago.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita al sistema poder aprobar un pago
Actor	Administrador.
Pre-condiciones	CU - 18 Enivar comprobante de Pagos.
Flujo normal	1. El actor solicita al sistema que quiere aprobar un pago. 2. El sistema muestra la pantalla con los datos del pago. 3. El actor solicita confimar pago. 4. El sistema solicita confirmación de la operación. 5. El actor confirma la operación. 6. El sistema muestra mensaje de que se realizo correctamente la operación.
Flujo alternativo	4.1 Si el actor introduce datos del comprobante incorrectos, el sistema lo notifica y vuelve al paso 2.
Post-condiciones	El usuario ha enviado el comprobante de pago.

Diagramas de Actividades - Gestión de ingresos/egresos: en las figuras 3.36, 3.373.38,3.39 muestran los diagramas de los casos de uso CU-16, CU-17, CU-18 y CU-19

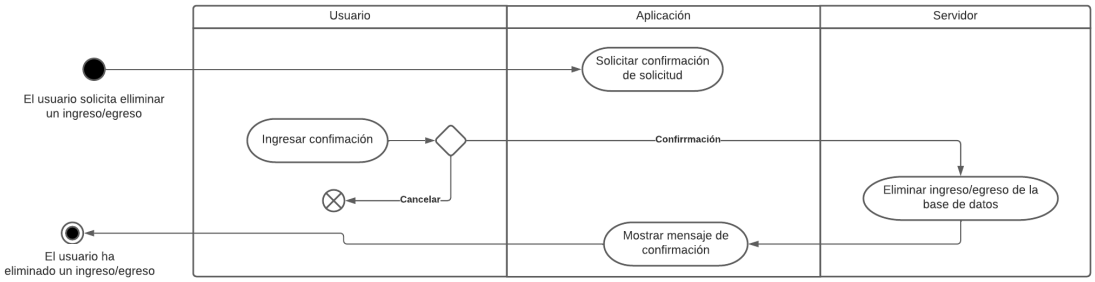


Figura 3.30: Diagrama de Actividad Caso de uso CU-15. Eliminar Ingreso/Egreso

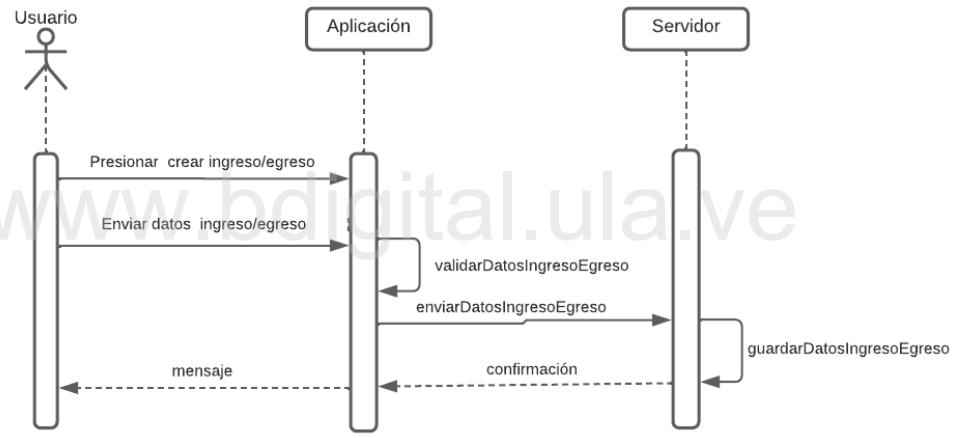


Figura 3.31: Diagrama de Secuencia Caso de uso CU-12. Crear Ingreso/Egreso

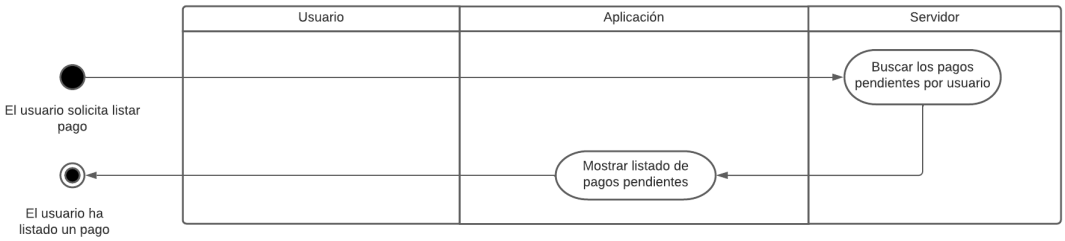


Figura 3.36: Diagrama de Actividad Caso de uso CU-16. Listar Pago

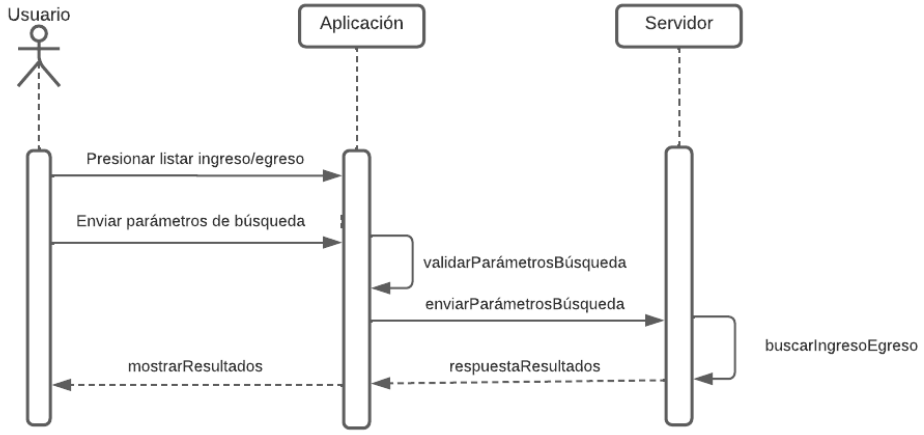


Figura 3.32: Diagrama de Secuencia Caso de uso CU-13. Listar Ingreso/Egreso

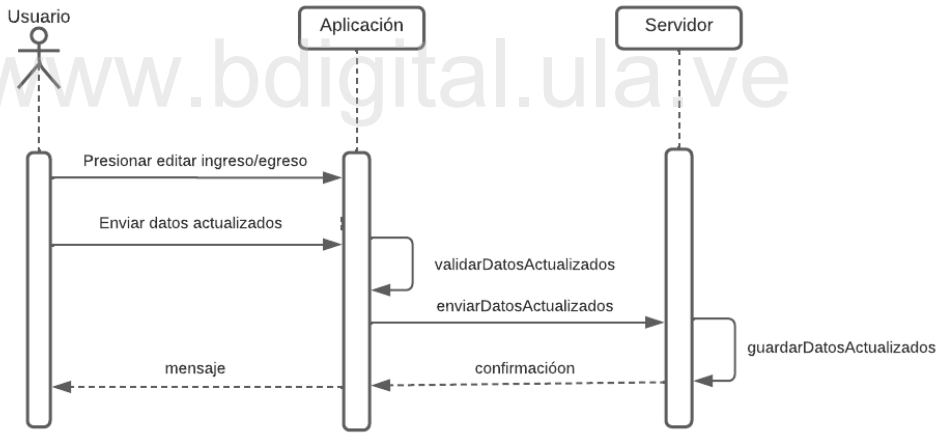


Figura 3.33: Diagrama de Secuencia Caso de uso CU-14. Editar Ingreso/Egreso

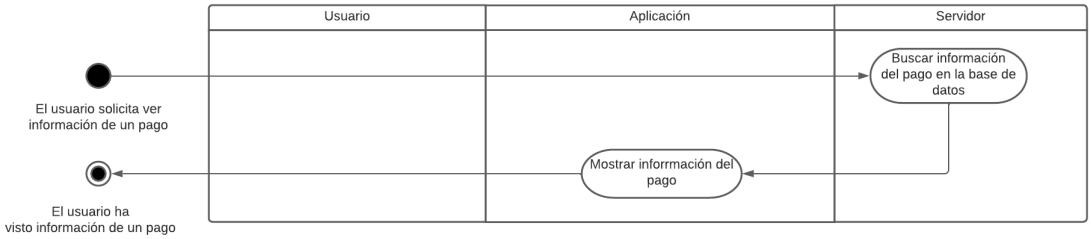


Figura 3.37: Diagrama de Actividad Caso de uso CU-17. Ver Pago

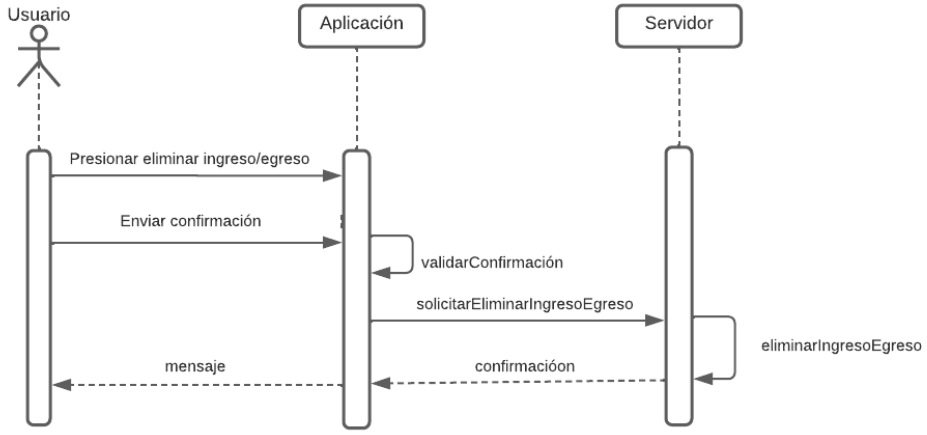


Figura 3.34: Diagrama de Secuencia Caso de uso CU-15. Eliminar Ingreso/Egreso

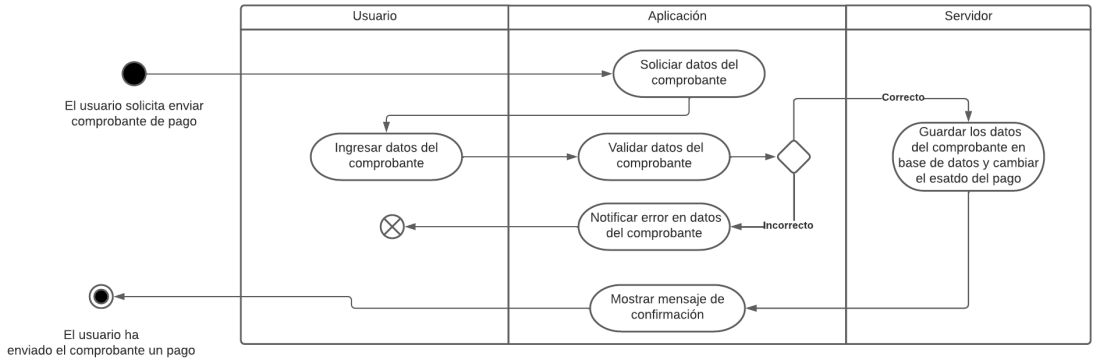


Figura 3.38: Diagrama de Actividad Caso de uso CU-18. Enivar comprobante de Pago

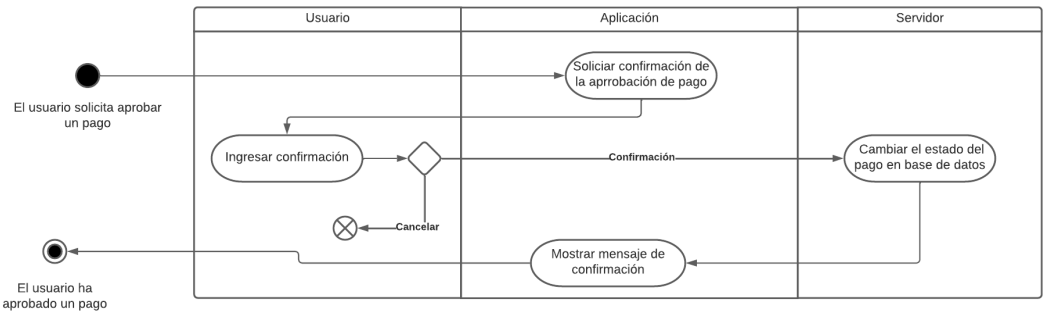


Figura 3.39: Diagrama de Actividad Caso de uso CU-19. Aprobar Pago

Diagramas Secuencia - Gestión de ingresos/egresos: en las figuras 3.40, 3.41,3.42,3.42 muestran los diagramas de secuencia de los casos CU-16, CU-17, CU-18 y CU-19.

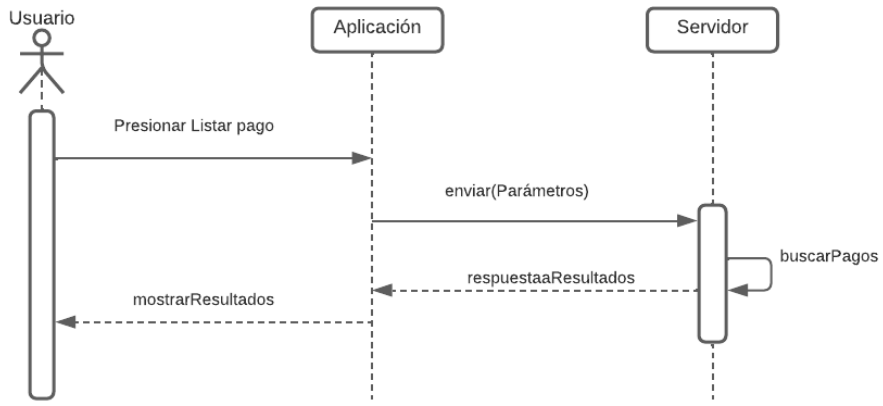


Figura 3.40: Diagrama de Secuencia Caso de uso CU-16. Listar Pago

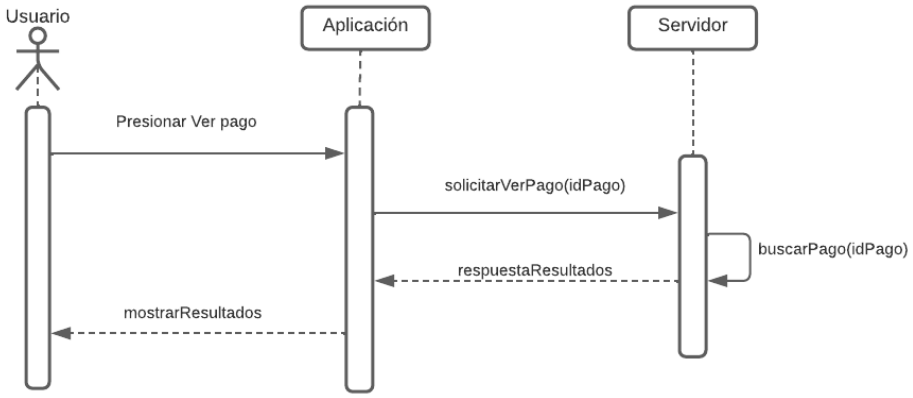


Figura 3.41: Diagrama de Secuencia Caso de uso CU-17. Ver Pago

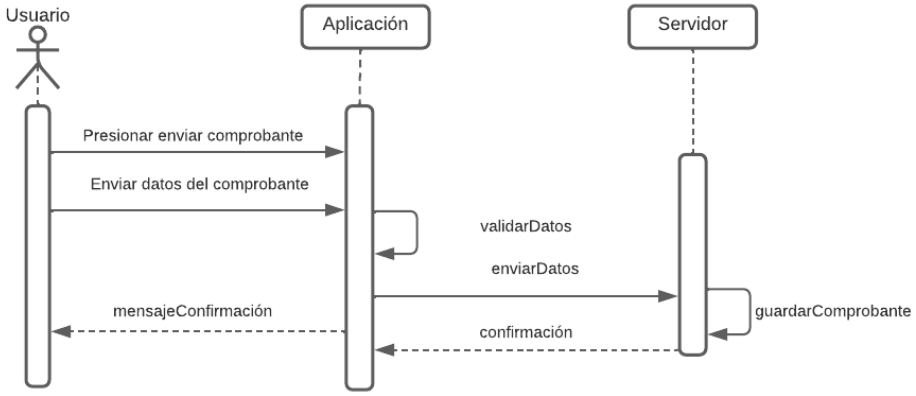


Figura 3.42: Diagrama de Secuencia Caso de uso CU-18. Enivar comprobante de Pago

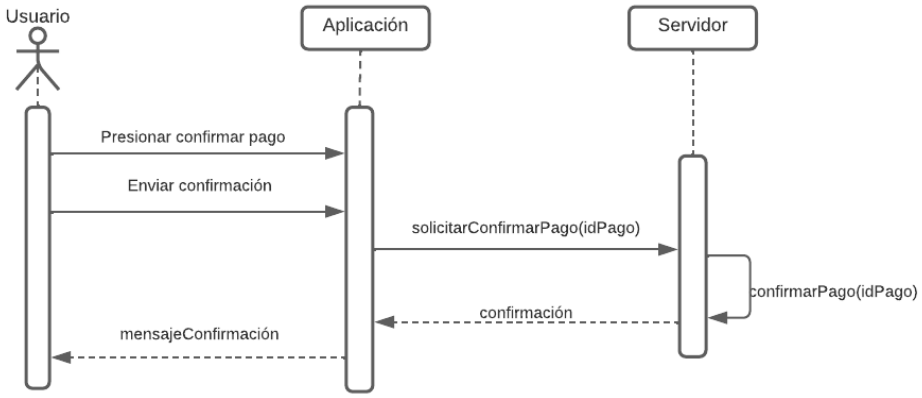


Figura 3.43: Diagrama de Secuencia Caso de uso CU-19. Aprobar Pago

3.2.2.6. Gestión de áreas comúes:

Los propietario pueden listar las áreas comúnes y hacer reservas de dichas áreas.

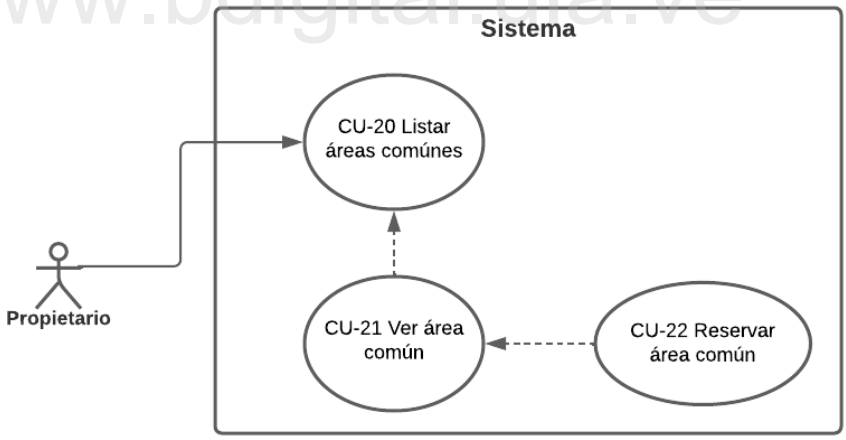


Figura 3.44: Diagrama de Caso de uso CU-06 Gestión de áreas comúes

Caso de Uso - Gestión de áreas comúes: en las tablas 3.24, 3.25,3.26 contienen la descripción textual detallada de los casos de uso mostrados en la Figura3.44.

Tabla 3.24: Descripción del Caso de Uso Listar áreas comúes.

Nombre	CU - 20 Listar áreas comúnes.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita al sistema listar las áreas comúnes del condominio.
Actor	Propietaario.
Pre-condiciones	Ninguna.
Flujo normal	1. El actor solicita al sistema que quiere listar las áreas comúnes del condominio. 2. El sistema busca en base de datos las áreas comúnes correspondiente al condominio. 3. El sistema muestra una pantalla con el listado de las áreas comúnes.
Flujo alternativo	Ninguno.
Post-condiciones	Ninguna.

Tabla 3.25: Descripción del Caso de Uso Ver áreas comúes.

Nombre	CU - 21 Ver área común.
Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita al sistema ver el área común del condominio.
Actor	Propietaario.
Pre-condiciones	CU - 20 Listar áreas comúnes.
Flujo normal	1. El actor solicita al sistema que quiere ver el área común del condominio. 2. El sistema busca en base de datos la información del área común. 3. El sistema muestra una pantalla con la información del área común.
Flujo alternativo	Ninguno.
Post-condiciones	Ninguna.

Tabla 3.26: Descripción del Caso de Uso Reservar área común.

Nombre	CU - 22 Reservar área común.

Continua en la siguiente página.

Tabla 3.26 – *Continuación de la página anterior.*

Descripción	Este caso de uso describe el proceso mediante el cual un usuario solicita al sistema reservar el área común.
Actor	Propietaario.
Pre-condiciones	CU - 21 Ver áreas comúnes.
Flujo normal	1. El actor solicita al sistema que quiere reserver el área común. 2. El sistema solicita datos de reservación. 3. El actor ingresa datos de reservación. 4. El sistema válida datos de reservación. 5. El sistema agrega la reservación y muestra mensaja de confirmación.
Flujo alternativo	4.1 Si el actor introduce datos de reservación incorrectos, el sistema lo notifica y vuelve al paso 2.
Post-condiciones	El actor agregó una reservvaación.

Diagramas de Actividades - Gestión de áreas comúes: en las figuras 3.45, 3.463.47,3.39 muestran los diagramas de los casos de uso CU-20, CU-21, CU-22

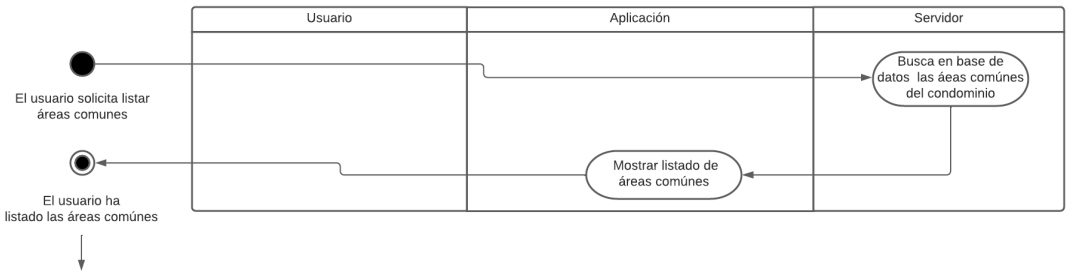


Figura 3.45: Diagrama de Actividad Caso de uso CU-20. Listar áreas comúnes

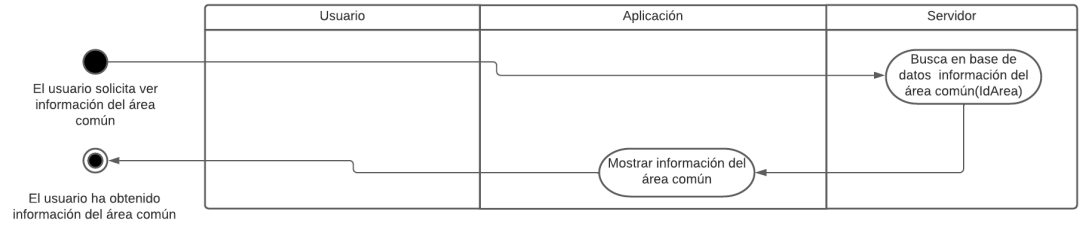


Figura 3.46: Diagrama de Actividad Caso de uso CU-21. Ver área común

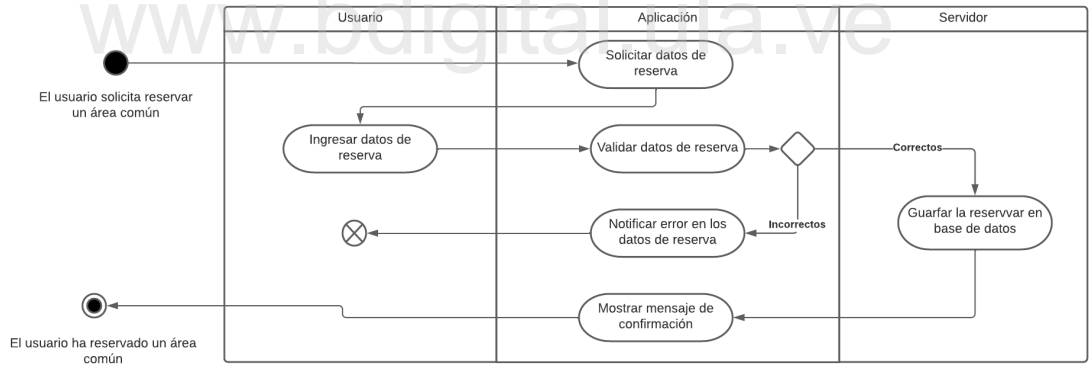


Figura 3.47: Diagrama de Actividad Caso de uso CU-22. Reservar áreas común

Diagramas Secuencia - Gestión de áreas comúes: en las figuras 3.40, 3.41,3.42,3.42 muestran los diagramas de secuencia de los casos CU-20, CU-21, CU-22.

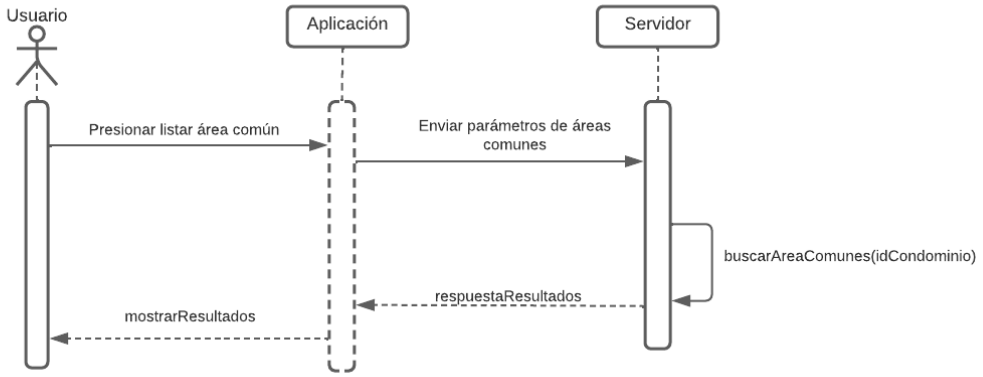


Figura 3.48: Diagrama de Secuencia Caso de uso CU-20. Listar áreas comúnes

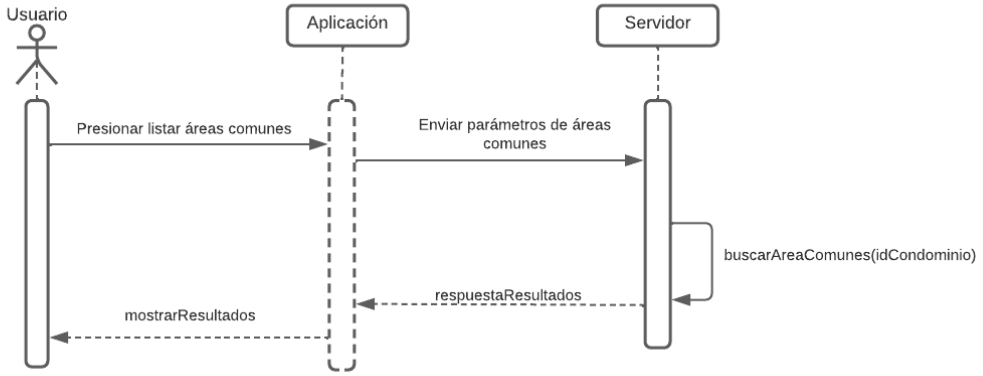


Figura 3.49: Diagrama de Secuencia Caso de uso CU-21. Ver área común

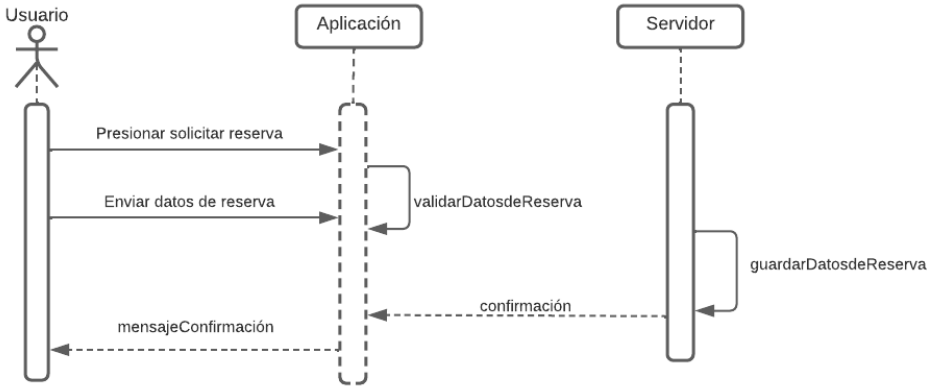


Figura 3.50: Diagrama de Secuencia Caso de uso CU-22. Reservar áreas común

3.2.3. Diseño de base de datos

En está sección se detalla las operaciones que se realizaron para obtener el diseño de la base de datos

3.2.3.1. Modelado Conceptual:

Consiste en la representación del UD (parte del mundo real que sequiere almacenar en la BD) en esquemas conceptuales E/R. Mediante los constructores del modelo E/R se recoge toda la semántica que puede obtenerse mediante la observación del UD o bien a partir de unas especificaciones textuales (esquemas descriptivos) que describan la información que debe contener la Base de Datos.

Representación del UD:

- Los administradores tienen un correo único,un número telefónico,un tipo de rol(administrador), nombres, apellidos. Para el ingreso de administradores se requiere el email/telefóno y una contraseña.

- Los propietarios tienen un correo único, un número telefónico, un tipo de rol(propietario),un número de apartamento unico para cada condominio, un

código de condominio, nombres, apellidos. Para el ingreso de propietarios se requiere el email/telefóno y una contraseña.

- Un administrador puede registrar vigilantes, un vigilante debe tener un correo único y una una contraseña.
- Para el ingreso de un vigilante se requiere el email/telefóno y una contraseña.
- Un administrador puede registrar uno o varios condominios. Un condominio tiene un código único que comparten todos los propietarios que pertenecen a un condominio especifico, un nombre, número de pisos, números de apartamentos y una dirección.
- Un administrador puede crear áreas comúnes por un condominio específico. Cada área común tiene un nombre, descripción y un código de condominio.
- Un administrador puede crear una tarea de mantenimiento. Cada tarea tiene un título, una descripción, un estado, un código de condominio.
- Un administrador puede crear noticias, una noticia debe tener un título y una descripción.
- Un administrador puede crear un pago. El pago debe contener un mes, un monto del pago, un apartamento, una descripción.
- Un administrador puede enviar notificaciones a los propietarios.
- Un administrador puede editar y eliminar propietarios.
- Un administrador puede registrar los gastos, un gasto debe llevar un monto del gasto, mes que se realizo el gasto y una descripción.
- Un propietaro puede enviar un comporbante de pago. El comprobante debe llevar una foto.
- Un administrador puede listar los comprobantes de pagos que esten en estado pendiente, por revisar y aprobados. Pueden cambiar el estado de los mismos.

- Un propietario puede registrar una visita. La visita debe tener nombres de la visita, número de apartamento.
- Un vigilante puede revisar las visitas programadas por los propietarios, a través de un código QR.
- Un vigilante puede registrar visitas no registradas.
- Un vigilante puede registrar paquetes que se entreguen en la portería. Cambiar estados de los paquetes.
- Un propietario puede revisar los paquetes registrados por los vigilantes.
- Un propietario puede listar áreas comúnes y hacer reservaciones de las mismas.

En las figuras 3.51, 3.52, 3.53 y 3.54 se muestra el diagrama ERE resultante de los procesos mas importantes.

3.2.3.2. Diseño Lógico:

Apartir del diagrama entidad-relación-extendido obtenido en el paso anterior, se comienza el diseño lógico considerando las siguientes reglas:

1. Cada entidad se transforma en una tabla y los atributos de dicha entidad en atributos de la tabla. Esto es, cada entidad genera una tabla, con sus mismos atributos, incluyendo las claves.
2. Las relaciones de muchos a muchos se transforman en tablas cuya clave estará formada por la clave primaria de las entidades relacionadas, convirtiéndose esta en una relación de muchos a muchos o de muchos a uno.
3. Las relaciones de uno a muchos propagan la clave principal de la entidad cuya cardinalidad es uno a la entidad de cardinalidad n.

Usuario

id	email	password	firstName	lastName	rol

Detalle de usuario

id	whatsapp	userId	numberApto	oneSignal	code

Pagos

id	total	mes	code	type	description	numberApto	status

Enviar comprobantes

id	photo	date

Condominios

id	name	code	userId	address	numberApto	numberFloor

Aparmentos

id	code	owner	status	numberApto

Tarea

id	code	userId	status	description

Noticas

id	code	title	description

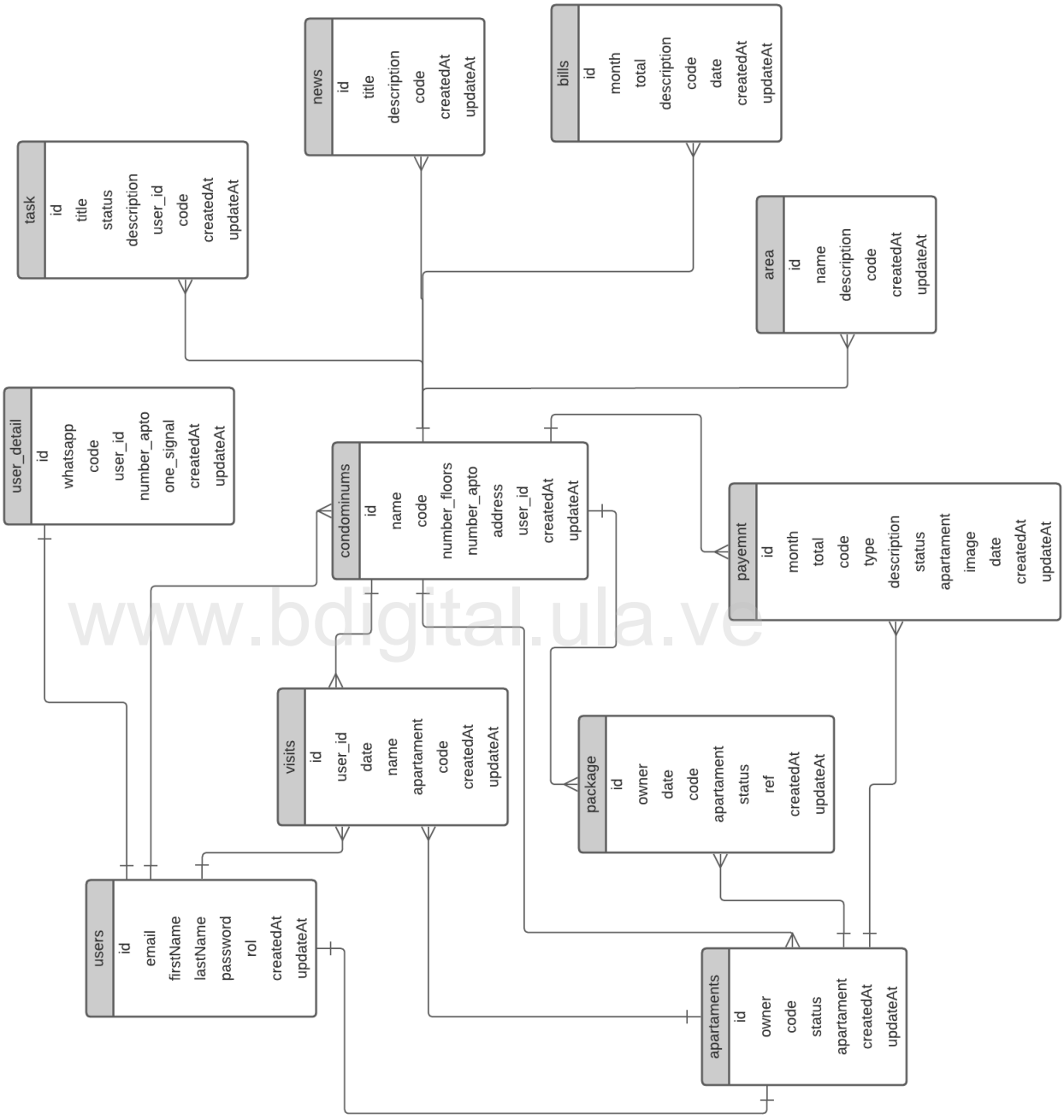


Figura 3.55: Diagrama UML del esquema físico de la base de datos.

3.2.3.3. Diseño Físico:

Despues de obtener el diseño lógico podemos contruir el el diseño físico, que se muestra en la figura 3.55. Para evitar usar tildes y carácteres especiales los nombres de las tablas fueron establecidos en inglés y plural; las relaciones entre cada tabla se representan a traves de lineas.

Area

id	name	code	description

Package

id	owner	code	date	apartament	status	ref

Pagos

id	month	code	total	description	date

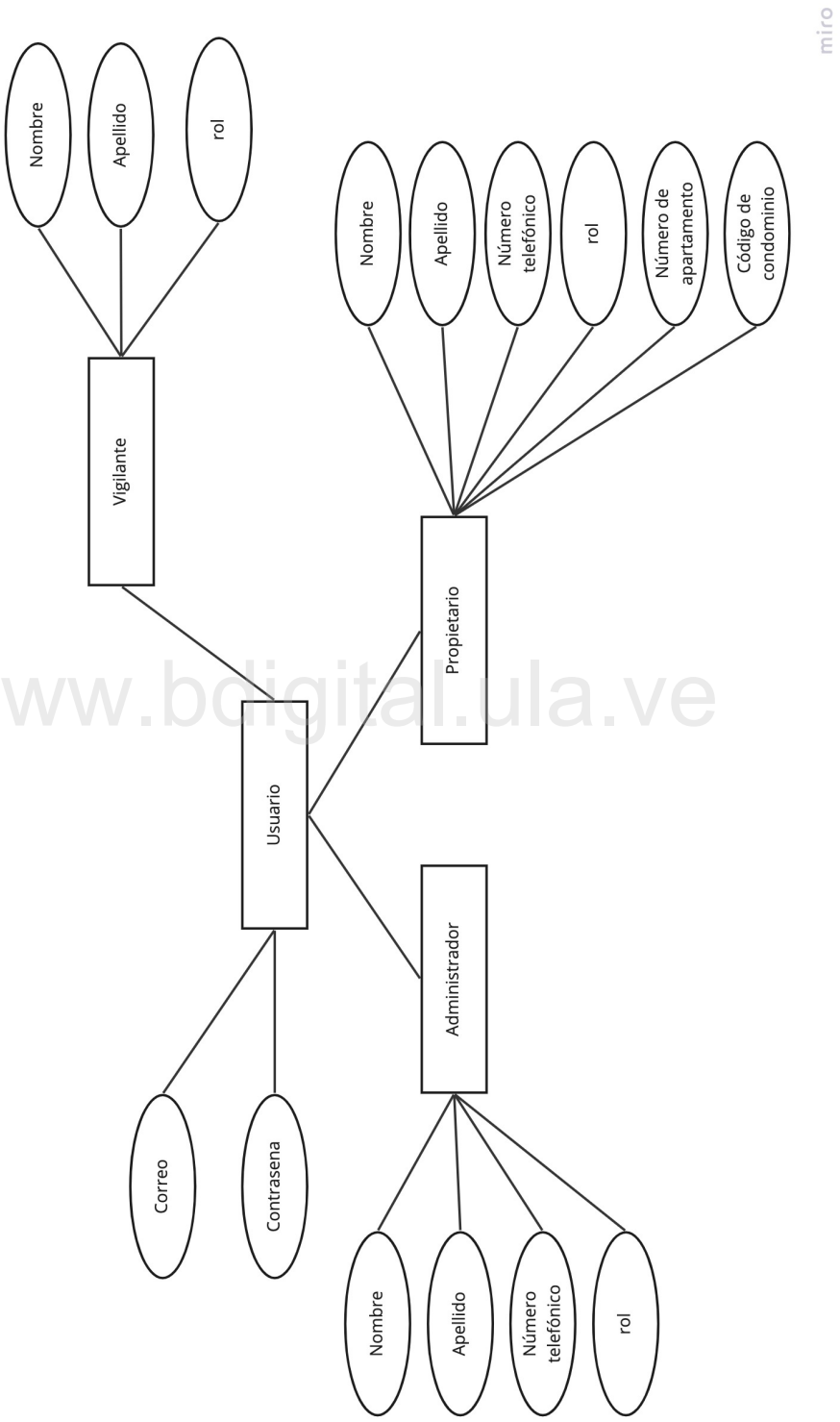


Figura 3.51: Diagrama entidad-relación-extendido de la base de datos (proceso 1).

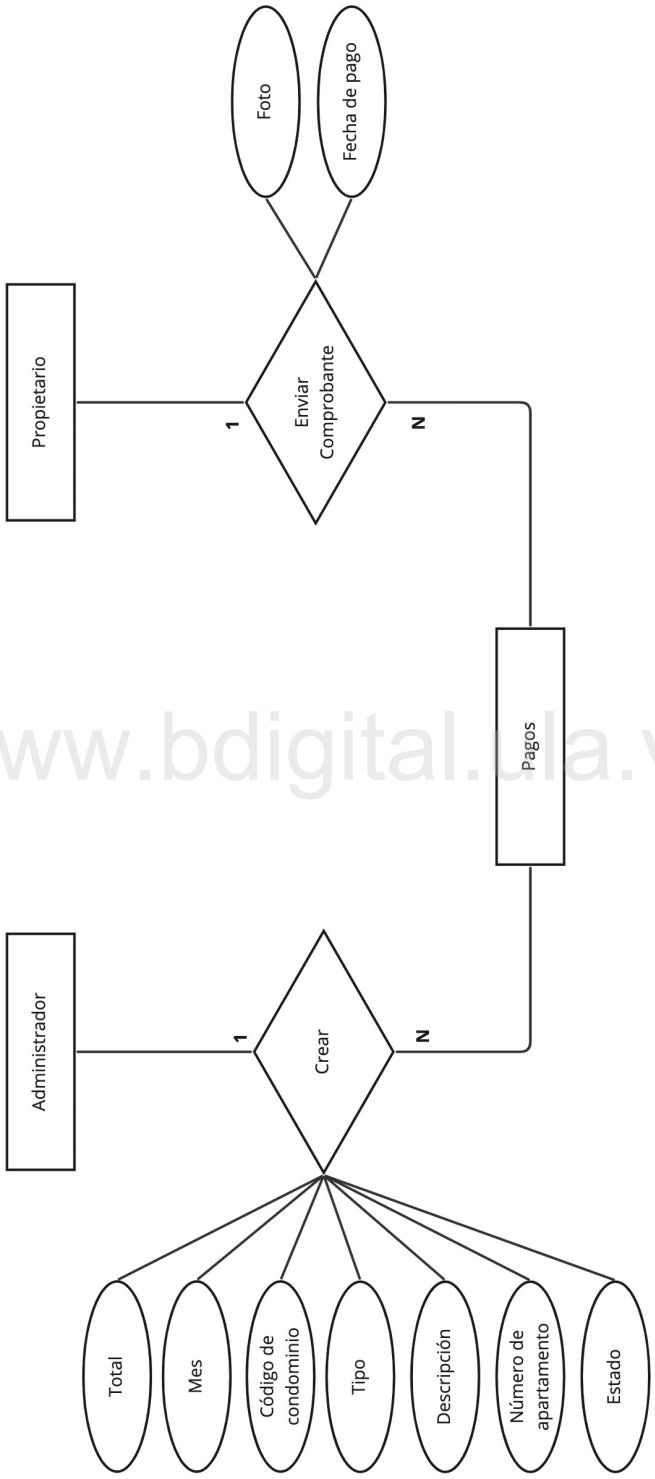


Figura 3.52: Diagrama entidad-relación-extendido de la base de datos (proceso 2).

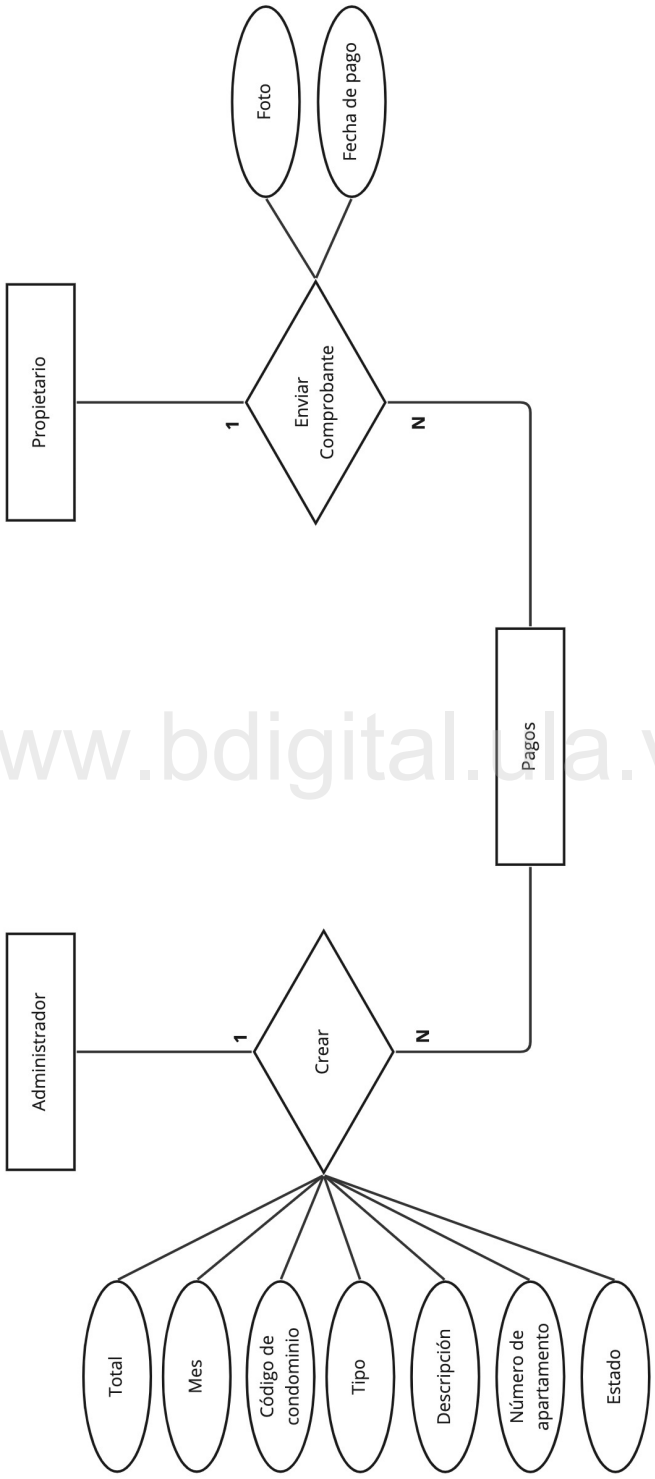


Figura 3.53: Diagrama entidad-relación-extendido de la base de datos (proceso 3).

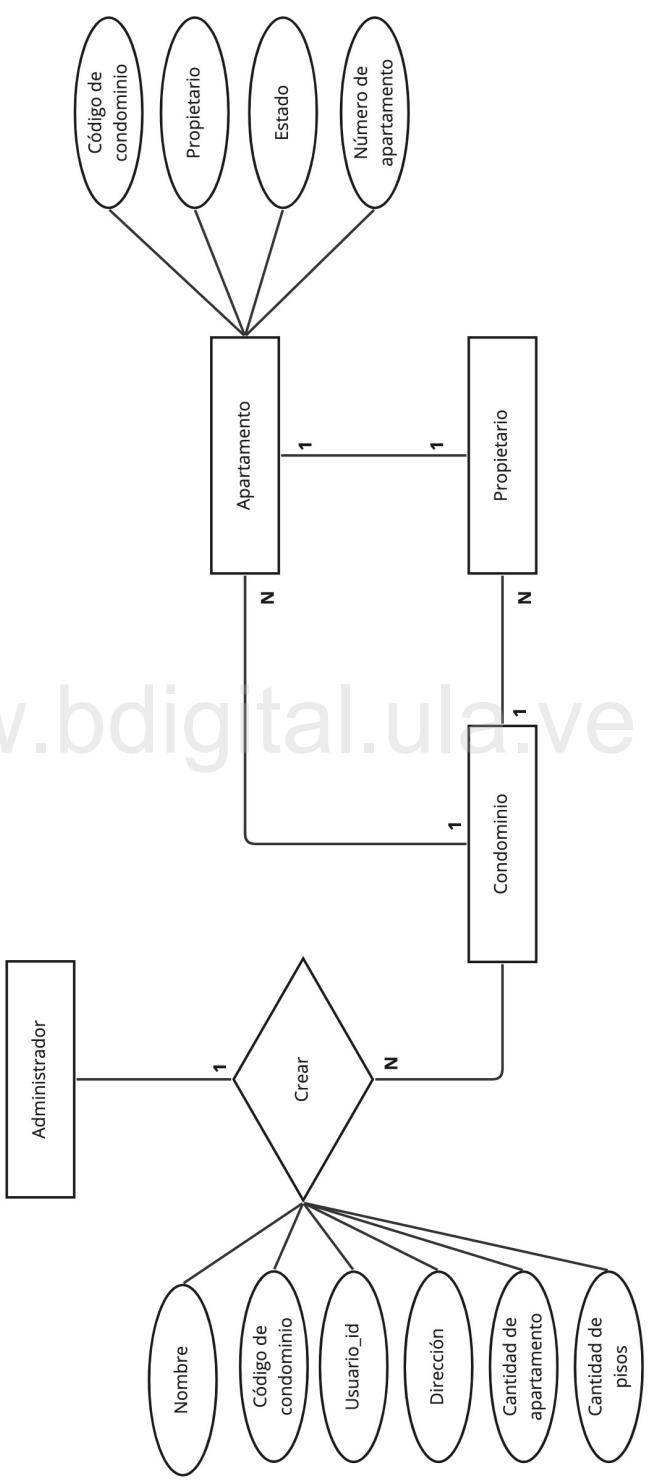


Figura 3.54: Diagrama entidad-relación-extendido de la base de datos (proceso 4).

Capítulo 4

Implementación y Pruebas

En este capítulo se detalla las 3 etapas restantes de las metodología Mobil-D que son producción, estabilización y pruebas. En la fase de producción de cada componente se hacian ciclos de tres días de programación, despues se continuaba con la fase de estabilización donde se verifico el correcto funcionamiento del componente y se creo la documentación necesaria. En la fase de pruebas se realizaron en emuladores y dispositivos reales;, estas pruebas incluían pruebas funcionales

4.1. Fase 3: Producción

Cuando se realizaba el diseño de una parte del sistema, se realizaba la codificación de la app y de la API REST.

4.1.1. Actividad 1: Codificación de la Aplicación

La codificación de la aplicación se llevó a cabo utilizando el Framework Ionic en su versión 6 para lo cual fue necesario instalar y configurar las siguientes tecnologías y herramientas:

- Ionic v6
- Npm 8.15.0
- Node v14.17.4
- Gradle v7.3.3

- Android Studio 4.2.1

4.1.1.1. Estructura de directorios:

Al generar el proyecto de Ionic se creó una estructura predefinida de carpetas y ficheros que nos permiten organizar el código de la aplicación.

- **node_modules/** Contiene todos los módulos de Ionic y dependencias que nos hemos instalado.
- **platforms/** Contiene el código espeífico de las plataformas móviles para las cuales se va a compilar, como por ejemplo Android.
- **plugins/** Contiene los plugins de Cordova instalados para las plataformas destino, los cuales se utilizan para añadir funcionalidad como el acceso a las características nativas del móvil.
- **resources/** Recursos específicos de las plataformas. Principalmente se utiliza para almacenar los iconos e imágenes de splashscreen que se utilizarán para cada plataforma.
- **src/** En esta carpeta se encuentra el código del proyecto.
  - **app/** Esta carpeta contiene los componentes principales del proyecto de Ionic:
    - **app.components.ts** Es el componente principal con la configuración de la aplicación que se carga al inicio.
    - **app.module.ts** El archivo contiene los servicios de contenido de la aplicación.
    - **app-routing.module.ts** El archivo donde se crean las rutas de la aplicación.
    - **news/**E n esta carpeta se encuentra los archivos para listar y crear una noticia de mantenimiento.
    - **task/** En esta carpeta se encuentra los archivos para listar, crear o cambiar el estado de una tarea de mantenimiento.
    - **login/** En esta carpeta se encuentra los archivos para iniciar sesión.

- **register/** En esta carpeta se encuentra los archivos para el registro.
- **packages/** En esta carpeta se encuentra los archivos para listar, crear o cambiar el estado de un paquete.
- **receipts/** En esta carpeta se encuentra los archivos para listar, crear o cambiar el estado de un pago.
- **balance/** En esta carpeta se encuentra los archivos para el balance de los gastos de la aplicación.
- **api/** En esta carpeta contiene los servicios de conexión para la API REST.
- **components/** Esta carpeta contiene todos los componentes reutilizables de la app (header, modal, botones...)
- **services/** Esta carpeta contiene servicios utilizados por la aplicación como SwalAlert y el Storage.

- **assets** Esta carpeta incluye las imágenes estáticas usadas en la aplicación.
- **theme/variables.scss** En este archivo se usa para los estilos css de la aplicación (por ejemplo colores).

4.1.1.2. Paquetes de terceros utilizados:

Para desarrollar la aplicación se utilizaron los siguientes paquetes de terceros:

- **cordova-plugin-camera** Este complemento define un navigator.cameraobjeto global, que proporciona una API para tomar fotografías y elegir imágenes de la biblioteca.
- **cordova-plugin-file-transfer** Este complemento le permite cargar y descargar archivos. El FileTransferobjeto proporciona una forma de cargar archivos mediante una solicitud HTTP POST o PUT de varias partes y descargar archivos.
- **onesignal-cordova-plugin** Es un servicio gratuito de correo electrónico, sms, notificaciones automáticas y mensajes integrados en la aplicación para aplicaciones móviles.
- **@techiediaries/ngx-qrcode** Este plugin se utilizá para generar fácilmente códigos QR dentro de sus aplicaciones Ionic.

- **phonegap-plugin-barcodescanner** Este plugin abre una vista de cámara y escanea automáticamente un código de barras, devolviéndole los datos.
- **@angular/common/http** Realiza solicitudes HTTP. Este servicio está disponible como una clase inyectable, con métodos para realizar solicitudes HTTP.
- **@ionic/storage-angular** Un módulo de almacenamiento de clave-valor simple para aplicaciones Ionic.
- **ng-lazyload-image** Una biblioteca súper pequeña para imágenes de carga diferida para aplicaciones Ionic/Angular sin dependencias.
- **sweetalert2** una librería de JavaScript para mostrar alertas y diálogos de confirmación con un diseño amigable con el usuario.

4.1.2. Actividad 2: Codificación del API REST

La codificación del API REST se llevó a cabo utilizando el lenguaje NodeJS en su versión 6 para lo cual fe necesario instalar y configurar las siguientes tecnologías y herramientas:

- Npm 8.15.0
- Node v14.17.4
- Nodemon v2.0.20

4.1.2.1. Estructura de directorios:

Cómo NodeJS no posee una estructura de archivo predefinidas, fue necesario establecer la estructura de archivos, esta se resume a continuación:

- **database/** Se encuentra la configuración para la conexión de la base de datos.
- **routes/** Contiene la configuración de las rutas del API REST.
- **models/** Contiene la definición de los modelos de las tablas definidas en la base de datos.
- **controllers/** Contiene las funciones a ejecutarse en las rutas definidas.

4.1.2.2. Paquetes de terceros utilizados:

Para el desarrollo del API REST se utilizaron los siguientes paquetes de terceros:

- **express/** Permite estructurar la aplicación de una manera ágil, nos proporciona funcionalidades como el enrutamiento, opciones para gestionar sesiones y cookies, etc.
- **express-validator/** Permite crear validaciones en las rutas.
- **sequilize/** Permite manipular varias bases de datos SQL de una manera sencilla.
- **onesignal-node/** Permite crear notificaciones push.
- **dotenv/** Permite configurar las variables de entorno.

4.1.2.3. Rutas/endpoints del API REST:

Las siguientes tablas contienen la lista de los endpoints del API REST.

Tabla 4.1: Rutas para operaciones de usuarios

Método:	Ruta:	Función:
POST	api/user/login	Inicio de sesión
POST	api/user/register	Registrar usuario.
GET	api/user/getUser	Obtener datos de un usuario
PUT	api/user/update/id	Actualizar datos un usuario
PUT	api/user/password	Actualizar contraseña de un usuario
PUT	api/user/delete/id	Eliminar un usuario.

4.2. Fase 5: Pruebas

En etapa se realizaron las distintas pruebas tanto de la aplicación como del API REST.

Tabla 4.2: Rutas para operaciones de comndominios

Método:	Ruta:	Función:
POST	api/condominium/create	Crear condominio.
POST	api/condominium/search	Buscar condominio.
GET	api/condominium/getUser	Obtener usuarios de un condominio.
PUT	api/condominium/all/id/page	Obtener condominios.
PUT	api/condominium/get/id	Obtener datos de un condominio.
GET	api/apartamente/code	Obtener los apartamentos de un condominio.

Tabla 4.3: Rutas para operaciones de área común

Método:	Ruta:	Función:
POST	api/area/create	Crear área común.
GET	api/area/get/id/page	Obtener las áreas comúnes de un condominio.

Tabla 4.4: Rutas para operaciones de noticias

Método:	Ruta:	Función:
POST	api/news/create	Crear noticia.
GET	api/news/get/all/id/page	Obtener las noticias de un condominio.
GET	api/news/get/id	Obtener información de una noticia.

Tabla 4.5: Rutas para operaciones de tarea de mantenimiento

Método:	Ruta:	Función:
POST	api/task/create	Crear tarea de mantenimiento.
PUT	api/task/update	Actualizar tarea de mantenimiento.
GET	api/task/code/page	Obtener tareas de mantenimiento de un condominio.

Tabla 4.6: Rutas para operaciones de pagos

Método:	Ruta:	Función:
POST	api/payment/create	Crear pago.
PUT	api/payment/update	Actualizar pago.
GET	api/payment/all/id/page/status	Obtener pagos de un condominio.
GET	api/payment/user/id/apartament/page	Obtener pagos de un usuario.

Tabla 4.7: Rutas para operaciones de paquetes

Método:	Ruta:	Función:
POST	api/package/create	Crear paquete.
POST	api/package/search	Buscar paquete.
PUT	api/package/update	Actualizar pago.
GET	api/package/get/code/page	Obtener paquetes.
GET	api/package/get/owner	Obtener paquete de un usuario.

Tabla 4.8: Rutas para operaciones de visitas

Método:	Ruta:	Función:
POST	api/visits/create	Crear visita.

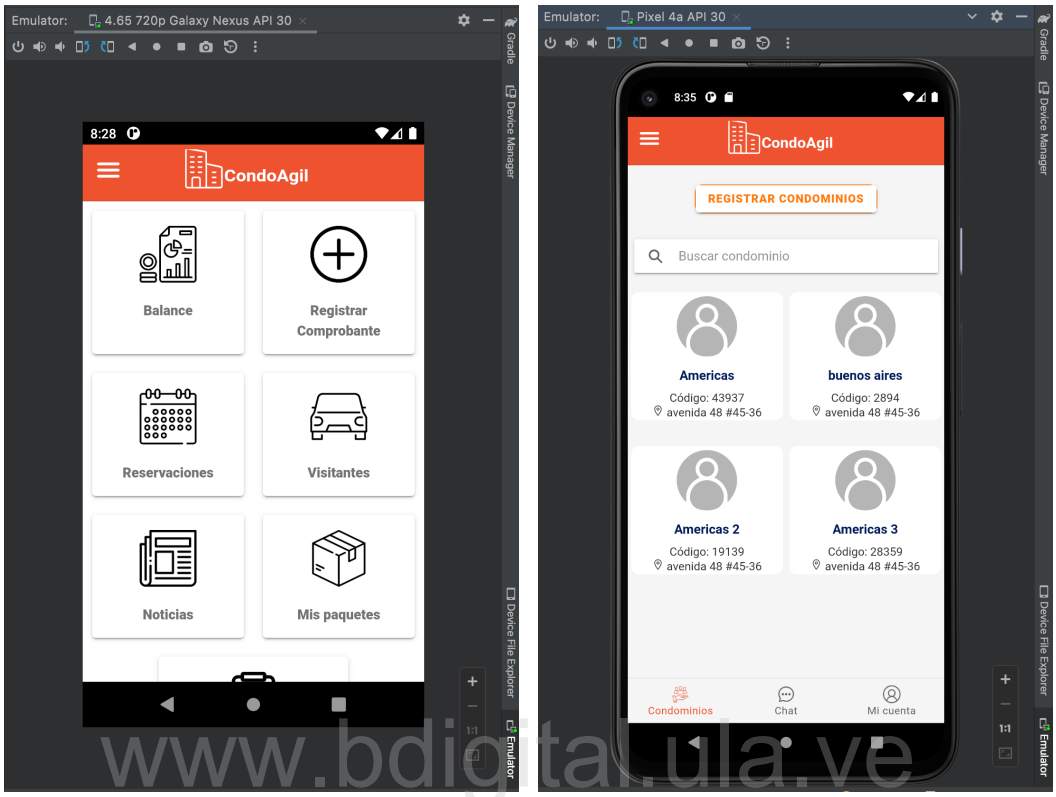
Tabla 4.9: Rutas para operaciones de gastos

Método:	Ruta:	Función:
POST	api/bills/create	Crear gastos.
GET	api/bills/get/code/page	Obtener gastos.

4.2.1. Actividad 1: Emulación y simulación:

Para realizar las pruebas de la aplicación móvil se utilizo el emulador Android Studio.

Para corroborar el correcto funcionamiento de la aplicación en dispositivos que no se tenían físicamente se utilizo el emulador Android Emulator que viene integrado en Android Studio. Esta prueba se realizó en dos dispositivo uno con arquitectura



(a) Galaxys Nexus. (b) Pixel 4a.

Figura 4.1: Pruebas en Android Emulator. Cuenta Administrador.

Galaxys Nexus API 30 y otro con arquitectura Pixel 4a API 31.

En la figura 4.1 se puede observar el comportamiento de la aplicación en ambos dispositivos.

4.2.2. Actividad 2: Dispositivos Reales:

La aplicación posee ciertas funcionalidades que solo pueden ser probadas en dispositivos reales, por lo cual se realizaron pruebas en un dispositivo físico cuyas características se listan en la tabla 4.10

En las figuras 4.2 y 4.3 se puede observar que las notificaciones push enviadas por el administrador son recibidas en el dispositvo real, támbien se detalla que la camara funciona correctamente para enviar un comprobante de pago y finalmente se

Tabla 4.10: Características del dispotivo fisíco Galaxys A30.

Características	Galaxys A30.
Procesador	OctaCore (Dual 2HGz + Hexa 1.7HGz).
Memoria RAM	4GB.
Pantalla	Super AMOLED de 6,4 pulgadas.
Sistema Operativo	Android 12.

muestra que la lectura del código QR se hace correctamente

4.2.3. Actividad 3: Pruebas Funcionales

A cada componente creado tanto en la aplicación como en el API REST se le aplicó un conjunto de pruebas unitarias. Para ejecutar dichas pruebas en la aplicación se instalo Karma.js, el cual se encarga de ejecutar los test que se configuraron en cada componente y así detectar cualquier fallo en la aplicación.

Para ejecutar estos test, se debe ejecutar el comando npm run test y en la figura 4.4 se observa que las scripts programados fueron existosos.

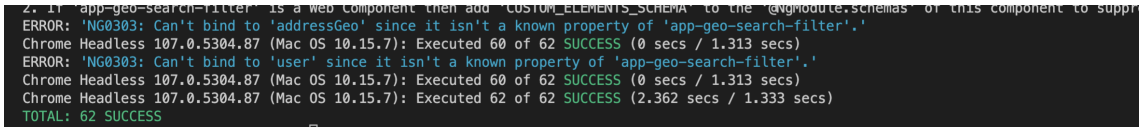
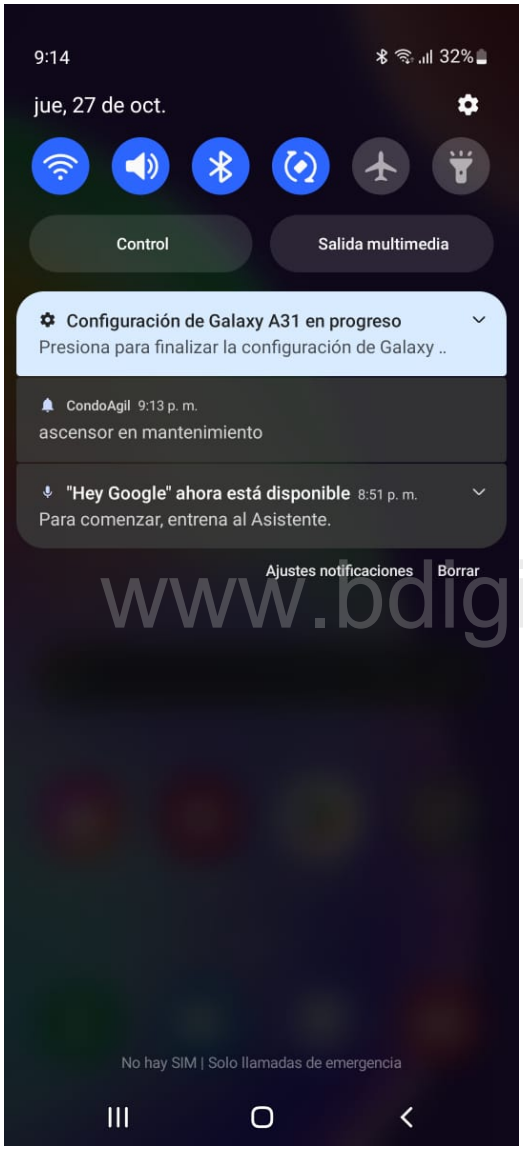


Figura 4.4: Pruebas automatizadas en la aplicación.

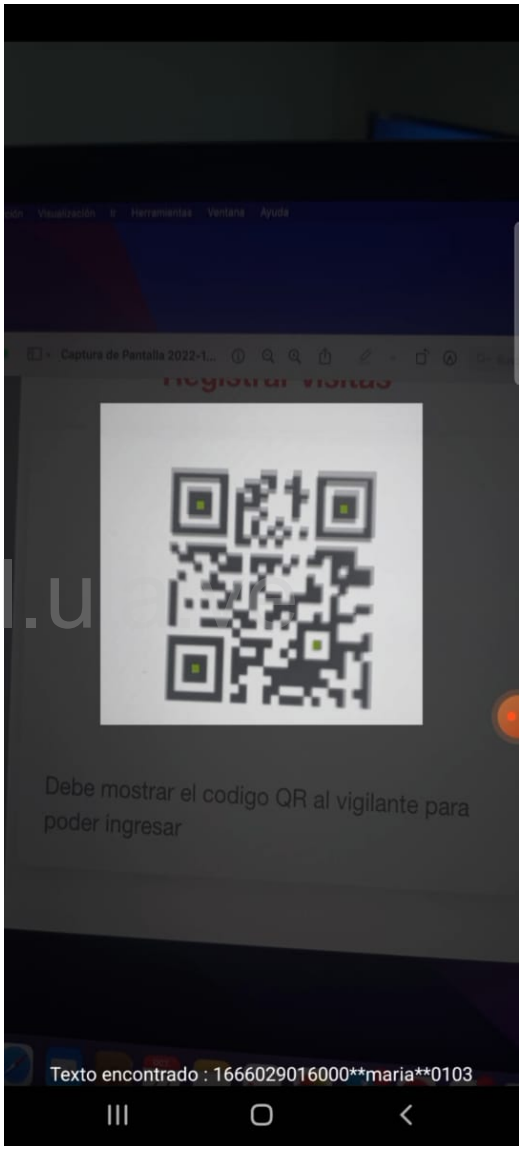
En el API REST se hicieron tanto pruebas manuales como pruebas automatizadas, para ello se utilizó la herramienta Postman, que permitió enviar peticiones al API y observar la respuesta devuelta por esta.

En la figura 4.5,4.6,4.7 se observa una de las pruebas manuales realizadas en Postman para el proceso de login.

En la figura 4.8,4.9,4.10 se muestra una de las pruebas manuales realizadas en Postman para el proceso de obtner los condominios de un administrador.

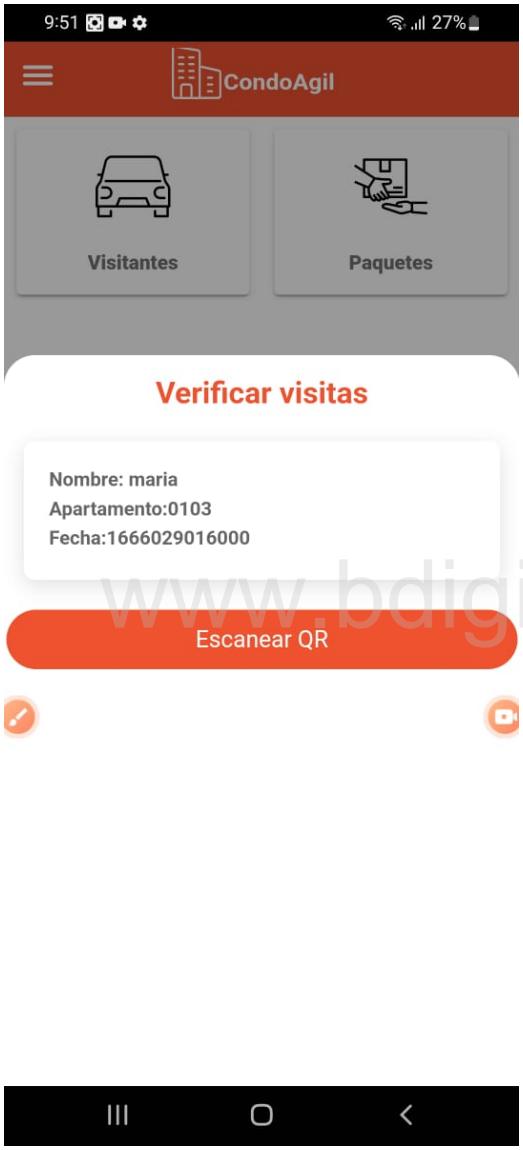


(a) Notificarión push.



(b) Lectura de código QR.

Figura 4.2: Pruebas en dispositivo físico Galaxys A30.



(a) Datos del código QR.

(b) Envio de comrobante

Figura 4.3: Pruebas en dispositivo físico Galaxys A30.

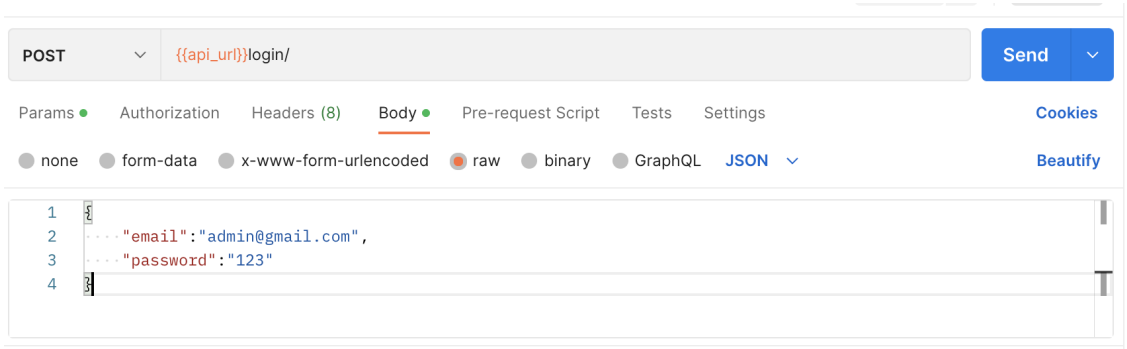


Figura 4.5: Pruebas manuales en Postman.Conguración de la petición.

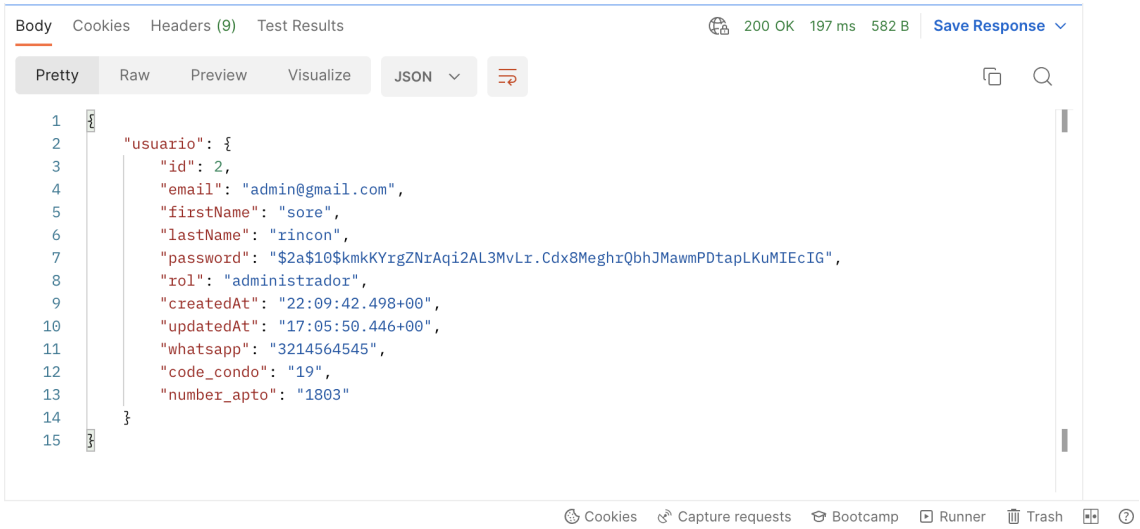


Figura 4.6: Pruebas manuales en Postman. Respuesta enviada por el API REST.

Figura 4.7: Pruebas manuales en Postman. Script de prueba.

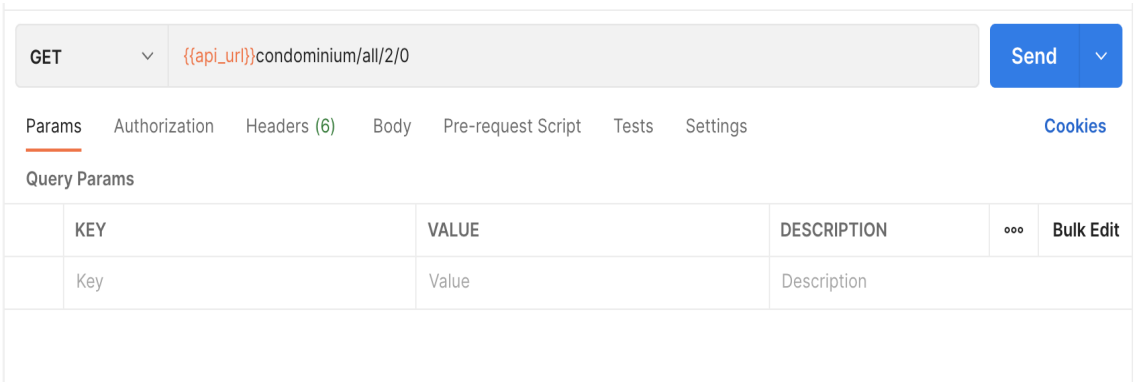


Figura 4.8: Pruebas manuales en Postman.Conguración de la petición.

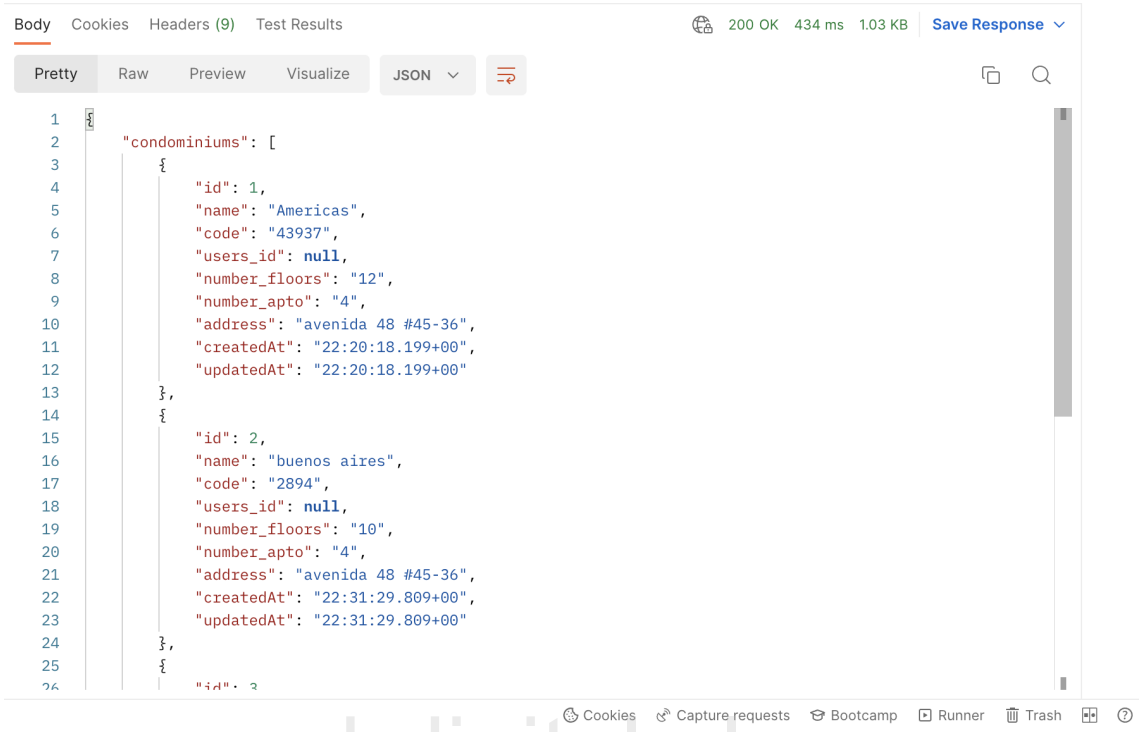


Figura 4.9: Pruebas manuales en Postman. Respuesta enviada por el API REST.

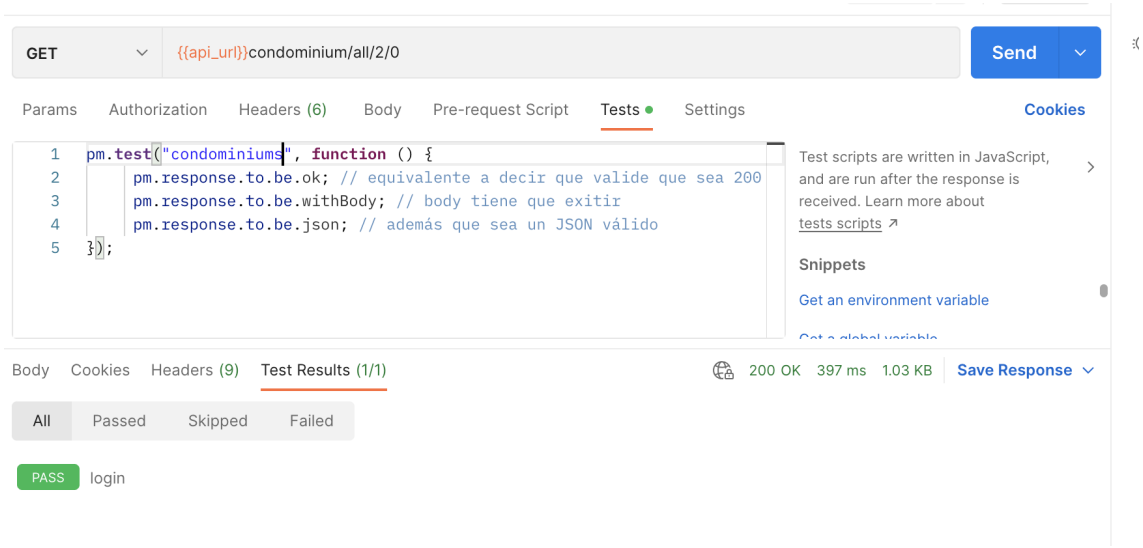


Figura 4.10: Pruebas manuales en Postman. Script de prueba.

Las pruebas automatizadas realizadas en Postman se configuraron con dos

iteracciones y un tiempo de retardo de 10ms. En la figura 4.11 se muestra el resumen de los resultados de las pruebas, el tiempo de ejecución y el código de respuesta.

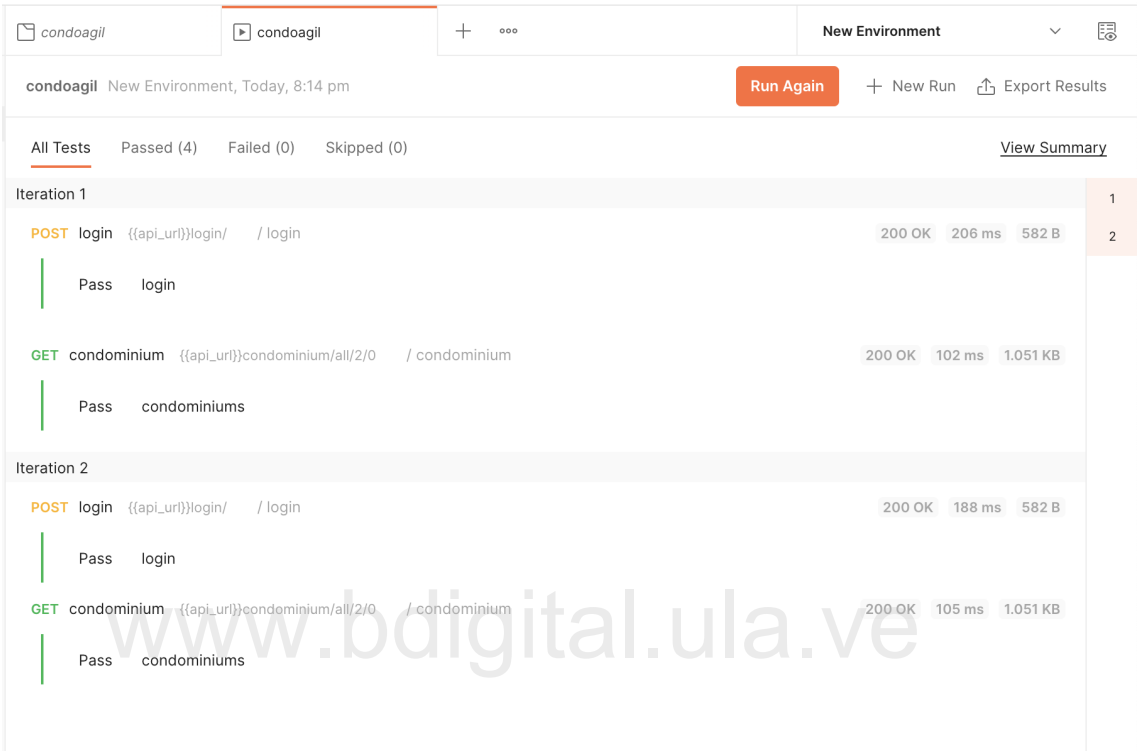


Figura 4.11: Pruebas automatizadas en Postman. Script de prueba.

Capítulo 5

Conclusiones y Recomendaciones

5.1. Conclusiones

En este proyecto se logró elaborar una aplicación móvil para para teléfonos inteligentes con SO Android y un API REST elaborada en NodeJS.

En conclusión, podemos decir que en este trabajo se lograron los siguientes resultados.

- Se realizo un analizo de la gestión y los procesos que ocurren dentro de la administración para obtener los requerimientos necesarios para el desarrollo de la aplicaicón.
- Se diseñaron los componentes y módulos a través de distintos diagramas que representaran la estructura y el comportamiento la aplicaicón.
- Se desarrolló una aplicación móvil que permita automatizar la gestión administrativa de condominios para teléfonos móvil en SO Android. Tambien se implemento un API REST.
-

5.2. Recomendaciones

- Configurar la aplicación para teléfonos móvil en SO iOS.
- Agregar pasarelas de pagos, para realizar pagos en líneas.

- Agregar un chat directo entre el administrador y propietario.
- Agregar módulo para los vigilantes puedan enviar notificaciones push.
- Permitir la recuperación de contraseña mediante un SMS o Whatsapp al número telefónico del usuario.

Referencias

Arquitectoit. Qué es postman? (Recuperado octubre 2021) Disponible en https://www.arquitectoit.com/postman/que-es-postman/.

Luis Huiza Boscán. Diseño e implementación de una aplicación móvil para el comercio electrónico de artículos de farmacia. 2020.

Alonso Rafael Cruz, Lilibeth Gopar Mecinas, Iván López López, y Edith Moya García. Investigación sobre android. 2020.

Congreso de la Repúplica. Ley de reforma parcial de la ley de propiedad horizontal. 1983. (Recuperado octubre 2021)Disponible en https://www.asambleanacional.gob.ve/storage/documentos/leyes/ley-de-ref-20220405162447.pdf.

DeveloperAndroid. Cómo ejecutar apps en android emulator. (Recuperado octubre 2021) Disponible en https://developer.android.com/studio/run/emulator?hl=es.

IonicFramework. About ionic. (Recuperado octubre 2021) Disponible en https://ionicframework.com/docs/v1/guide/preface.html.

Edwin Vinicio Ramos Moreno. Desarrollo de una aplicación movil para campos santos con acceso a una base de datos mediante quick response codes. 2022.

Cristian Muñoz. Aplicación de la metología mobile-d en el desarrollo de una app móvil para gestionar citas médicas del centro jel riobamba. 2021.

NodeJS. Acerca de node.js. (Recuperado octubre 2021) Disponible en https://nodejs.org/es/about/.

OneSignal. Documentación onesignal. (Recuperado octubre 2021)Disponible en `https://documentation.onesignal.com/docs`.

Angel Ortiz. Desarrollo e implementación de una aplicación móvil informativa para los estudiantes del instituto superior tecnológico primero de mayo. 2020.

Bryam Barrera Pedro Illaisaca. Desarrollo de una aplicación móvil y web para la inspección de juntas de agua potable usando servicios de geolocalización y almacenamiento en la nube. 2022.

Francisco Camazón Rodriguez. Desarrollo de una aplicación móvil de ayuda al estudio de la asignatura redes y servicios telemáticos. 2020.

Robert Ramírez Vique. Métodos para el desarrollo de aplicaciones móviles. 2018.