



UNIVERSIDAD
CATÓLICA
DE CUENCA

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

**FACULTAD DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN
E INNOVACIÓN TECNOLÓGICA**

CARRERA DE SOFTWARE

**Diseño e implementación de una plataforma web para la oferta y contratación de
servicios independientes con verificación de proveedores**

**PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO DE SOFTWARE**

AUTOR: JUAN CARLOS GUAMÁN TITUAÑA

DIRECTORA: ING. DIANA XIMENA POMA JAPÓN, MGS.

CUENCA - ECUADOR

2026

DIOS, PATRIA, CULTURA Y DESARROLLO



UNIVERSIDAD CATÓLICA DE CUENCA
Comunidad Educativa al Servicio del Pueblo
**FACULTAD DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN
E INNOVACIÓN TECNOLÓGICA**

CARRERA DE SOFTWARE

**Diseño e implementación de una plataforma web para la oferta y contratación de servicios
independientes con verificación de proveedores**

**PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO DE SOFTWARE**

AUTOR: JUAN CARLOS GUAMÁN TITUAÑA
DIRECTORA: ING. DIANA XIMENA POMA JAPÓN, MGS.

CUENCA - ECUADOR
2026
DIOS, PATRIA, CULTURA Y DESARROLLO



Universidad
Católica
de Cuenca

DECLARATORIA DE AUTORÍA Y RESPONSABILIDAD

Declaratoria de Autoría y Responsabilidad

Juan Carlos Guamán Tituaña portador(a) de la cédula de ciudadanía N° 0107190977. Declaro ser el autor de la obra: "Diseño e implementación de una plataforma web para la oferta y contratación de servicios independientes con verificación de proveedores", sobre la cual me hago responsable sobre las opiniones, versiones e ideas expresadas. Declaro que la misma ha sido elaborada respetando los derechos de propiedad intelectual de terceros y eximo a la Universidad Católica de Cuenca sobre cualquier reclamación que pudiera existir al respecto. Declaro finalmente que mi obra ha sido realizada cumpliendo con todos los requisitos legales, éticos y bioéticos de investigación, que la misma no incumple con la normativa nacional e internacional en el área específica de investigación, sobre la que también me responsabilizo y eximo a la Universidad Católica de Cuenca de toda reclamación al respecto.

Cuenca, 31 de julio de 2026

F: 

Juan Carlos Guamán Tituaña

C.I. 0107190977

CERTIFICADO DEL TUTOR

Yo, Diana Ximena Poma Japón, certifico que el presente trabajo de investigación, con el título "**Diseño e implementación de una plataforma web para la oferta y contratación de servicios independientes con verificación de proveedores**", fue desarrollado por Juan Carlos Guamán Tituaña con número de cédula 0107190977, bajo mi supervisión.



F:

Diana Ximena Poma Japón

C.I. 1104101710

Dedicatoria

A Jehová Dios, por darme la vida y la oportunidad de seguir aprendiendo y construyendo cada día, y por acompañarme en cada etapa de este camino.

A mi abuela, por su apoyo incondicional en todo lo que emprendo. Ha afrontado grandes dificultades en la vida y, a pesar de todo, sigue esforzándose por mejorar día con día; eso es algo que yo también quiero llegar a ser.

A mi padre, por ser un ejemplo de trabajo, constancia y honestidad, y por toda la ayuda que siempre me ha brindado; es el tipo de persona que aspiro poder ser.

A mi madre, por su fortaleza para salir adelante a pesar de las dificultades y por haber estado siempre a mi lado, sin condiciones.

Este trabajo es también un reflejo de lo que ustedes han sembrado en mí.

Juan Carlos Guamán Tituaña

Agradecimiento

A la Ing. Diana Ximena Poma Japón, MGS., directora de este trabajo de titulación, por su acompañamiento constante, su disposición y sus valiosos aportes a lo largo de todo el proceso de desarrollo de esta tesis.

Al Ing. Milton Campoverde Molina, PhD., director de la Carrera de Software, por su dedicación y entrega en las clases que impartió a lo largo de mi formación académica, y por su compromiso con la calidad educativa de la carrera.

A la Universidad Católica de Cuenca, por brindarme los espacios, los recursos y la formación necesarios para desarrollar este proyecto.

Juan Carlos Guamán Tituaña

Resumen

Durante el proyecto se analizó cómo se ofrecen y buscan servicios independientes en Cuenca. Se identificó que muchas personas recurren a recomendaciones, grupos de mensajería y anuncios públicos, medios que no siempre ofrecen información suficiente sobre quién presta el servicio. Por ello se desarrolló ServiLink, una plataforma web para publicar y contratar servicios independientes con verificación de proveedores. La investigación corresponde a un enfoque aplicado de tipo tecnológico, orientado a la construcción de una solución de software; el desarrollo del producto se organizó con el marco ágil Scrum en siete sprints de dos semanas. En el análisis de requerimientos se establecieron 21 requerimientos funcionales y 19 no funcionales, documentados en un SRS y en veinte diagramas UML que permitieron definir la arquitectura, la base de datos y la interfaz. La plataforma se construyó con Node.js, Express y MongoDB en el servidor, y React.js, Vite y Tailwind CSS en el cliente. Se implementó autenticación por roles, verificación documental de proveedores, publicación y búsqueda de anuncios, gestión de solicitudes con propuesta de citas y notificaciones automáticas mediante la WhatsApp Business Cloud API. El sistema se validó con 23 escenarios de prueba funcionales, con 23 PASS y 0 FAIL; es decir, el 100 % de los casos de prueba ejecutados resultó satisfactorio, y se desplegó en producción mediante Railway. Los resultados muestran que ServiLink cumple con los objetivos planteados y constituye una contribución potencial a la formalización de la contratación de servicios independientes en Cuenca, cuyo efecto real deberá evaluarse en trabajos posteriores con datos de adopción y uso.

Palabras clave: *plataforma web, verificación de proveedores, servicios independientes, desarrollo de software, metodología ágil*

Abstract

During the project, how independent services are offered and demanded in Cuenca will be analyzed. It was discovered that many people rely on recommendations, messaging groups, and public advertisements, which do not always provide enough information about service providers. Therefore, ServiLink, a web platform for posting and hiring independent services with provider verification, was developed. This research adopted a quantitative approach, and the development was organized using Scrum in seven two-week sprints. During the requirements analysis, 21 functional and 19 non-functional requirements were established and documented in an SRS and 20 UML diagrams to define the architecture, database, and interface. The platform was built using Node.js, Express, and MongoDB on the server side, and React.js, Vite, and Tailwind CSS on the client side. Role-based authentication, documentary verification of providers, advertisement posting and search, request management with appointment proposals, and automated notifications via the WhatsApp Business Cloud API were implemented. The system was validated with 24 functional test cases, yielding 24 PASS and 0 FAIL results (100% test coverage), and was deployed to production using Railway. The results show that ServiLink meets the objectives and can contribute to reducing the informal hiring of independent services in Cuenca.

Keywords: *Web platform, provider verification, independent services, software development, agile methodology*

Índice

Introducción	1
Capítulo I	2
1.1. Planteamiento del problema	2
1.2. Justificación.....	3
1.3. Objetivos.....	5
1.4. Preguntas de investigación	6
1.5. Alcance	7
1.6. Metodología.....	8
Capítulo II	9
2.1. Estado del arte	9
Comparación de antecedentes y plataformas de referencia.....	13
2.2. Marco teórico referencial.....	14
Capítulo III.....	20
3.1. Enfoque metodológico.....	20
3.2. Flujo de trabajo.....	21
3.3. Análisis de requerimientos	22
3.3.1. Actores del sistema.....	22
3.3.2. Requerimientos funcionales	23
3.3.3. Requerimientos no funcionales	24

3.4. Diseño de la arquitectura	26
3.5. Diseño de la base de datos.....	28
3.6. Casos de uso — descripción narrativa.....	31
3.7. Descripción de los diagramas UML	33
3.8. Restricciones técnicas del sistema.....	42
3.9. Marco de desarrollo de software	43
Capítulo IV.....	50
4.1. Implementación del backend.....	50
4.2. Implementación del frontend.....	70
4.3. Despliegue en producción	72
4.4. Pruebas funcionales	73
4.5. Análisis de resultados	78
4.6. Validación con usuarios finales.....	80
4.7. Conclusiones.....	84
4.8. Recomendaciones	86
4.9. Trabajos futuros.....	87
Referencias.....	89
Anexos	94
Anexo A. Matriz completa de requerimientos funcionales	94
Anexo B. Matriz completa de requerimientos no funcionales	95

Anexo C. Diagramas de secuencia complementarios.....	97
Anexo D. Matriz de trazabilidad entre requerimientos, casos de uso, diagramas y pruebas	103
Anexo E. Autorización de publicación en el Repositorio	105

Índice de tablas

Tabla 1	23
Tabla 2	25
Tabla 3	31
Tabla 4	42
Tabla 5	76
Tabla 6	82
Tabla 7 <i>Matriz completa de requerimientos funcionales de ServiLink</i>	94
Tabla 8 <i>Matriz completa de requerimientos no funcionales de ServiLink</i>	96

Índice de ilustraciones

Figura 1	22
<i>Plan de sprints del proyecto ServiLink — 7 iteraciones de dos semanas con entregables verificables.</i>	22
Figura 2	26
<i>Diagrama de componentes</i>	26
Figura 3	29
<i>Diagrama de clases</i>	29
Figura 4.....	34
<i>Casos de uso — Autenticación</i>	34
Figura 5.....	34
<i>Casos de uso — Rol dual</i>	34
Figura 6	35
<i>Casos de uso — Búsqueda y perfil</i>	35
Figura 7	35
<i>Casos de uso — Solicitudes y citas</i>	35
Figura 8.....	35
<i>Casos de uso — Gestión de anuncios</i>	35
Figura 9	36
<i>Casos de uso — Verificación de proveedores</i>	36
Figura 10.....	36
<i>Casos de uso — Panel de administración</i>	36
Figura 11	37

<i>Casos de uso — Notificaciones WhatsApp</i>	37
Figura 12	38
<i>Diagrama de secuencia — Registro de proveedor: validación de cédula módulo 10, Multer, cuenta con verificationStatus pendiente</i>	38
Figura 13	39
<i>Diagrama de secuencia — Contratación completa de servicio: búsqueda, solicitud, notificación WhatsApp y aceptación</i>	39
Figura 14	39
<i>Diagrama de secuencia — Verificación de proveedor: aprobación y rechazo con notificación WhatsApp</i>	40
Figura 15	41
<i>Diagrama de secuencia — Upgrade de rol: validación de cédula, notificación al admin y actualización de roles[] (RF-15, RF-16)</i>	41
Figura 16	43
<i>Secuencia de los siete sprints del proyecto ServiLink con sus entregables verificables al cierre de cada iteración.</i>	43
Figura 17	52
<i>Swagger UI — Grupos Usuarios, Upgrade de rol y Solicitudes de servicio</i>	52
Figura 18	52
<i>Swagger UI — Grupos Auth, Anuncios y Administración con endpoints expandidos</i>	52
Figura 19	52
<i>RF-01 — Registro exitoso de cliente María López. Respuesta HTTP 201 Created con token JWT y roles: ["cliente"]</i>	52

Figura 20	53
<i>RF-02 Escenario A — Cédula inválida (0102030405): el sistema retorna HTTP 400 Bad Request.....</i>	<i>53</i>
Figura 21	53
<i>RF-02 Escenario B — Cédula duplicada (0107190977 ya registrada): el sistema retorna HTTP 409 Conflict.....</i>	<i>53</i>
Figura 22	54
<i>RF-02 Escenario C — Registro exitoso de Carlos Pérez (cédula 1706090113): HTTP 201 con verificationStatus: "pendiente"</i>	<i>54</i>
Figura 23	56
<i>GET /users/me — Perfil del usuario autenticado. El campo password no aparece en la respuesta (HTTP 200).....</i>	<i>56</i>
Figura 24	56
<i>PATCH /users/me — Actualización de nombre y teléfono. El campo updatedAt se actualiza automáticamente (HTTP 200).....</i>	<i>56</i>
Figura 25	58
<i>RF-05 — Control de acceso: cliente intenta subir documentos y recibe HTTP 403 Forbidden</i>	<i>58</i>
Figura 26	58
<i>RF-06 — Admin consulta verificaciones pendientes. Carlos Pérez con verificationStatus: "pendiente" (HTTP 200)</i>	<i>58</i>
Figura 27	58

<i>RF-07 Escenario A — Acción inválida: campo estado no permitido retorna HTTP 400 con lista de valores aceptados</i>	58
Figura 28	59
<i>RF-07 Escenario B — Verificación aprobada: verificationStatus cambia a "aprobado" (HTTP 200). Log WhatsApp en consola</i>	59
Figura 29	61
<i>RF-08 — Listado público paginado de anuncios activos sin autenticación (HTTP 200)</i>	61
Figura 30	61
<i>RF-19 Escenario A — Cliente intenta crear anuncio y recibe HTTP 403 Forbidden</i>	61
Figura 31	61
<i>RF-19 Escenario B — Carlos (proveedor verificado) crea anuncio: HTTP 201 Created con estado: "activo"</i>	61
Figura 32	64
<i>RF-10 Escenario 1 — María envía solicitud a Carlos: HTTP 201 Created, estado: "pendiente". Notificación WhatsApp al proveedor</i>	64
Figura 33	64
<i>RF-11 Escenario 2a — Cliente intenta aceptar solicitud propia: HTTP 403 Forbidden</i>	64
Figura 34	64
<i>RF-11 Escenario 2b — Administrador intenta aceptar: HTTP 403 Forbidden</i>	64
Figura 35	65
<i>RF-11 Escenario 2c — Carlos (proveedor dueño) acepta: HTTP 200 OK, estado: "aceptada"</i>	65
Figura 36	65

<i>RF-11 Escenario 4 — Carlos propone reagendamiento: HTTP 200, estado: "reagendada".</i>	
<i>Notificación WhatsApp al cliente.....</i>	<i>65</i>
<i>Figura 37</i>	<i>66</i>
<i>RF-13 Escenario 5a — María acepta el reagendamiento: HTTP 200, estado vuelve a</i>	
<i>"aceptada".....</i>	<i>66</i>
<i>Figura 38</i>	<i>68</i>
<i>RF-17 — Dashboard de estadísticas del administrador en tiempo real: 5 usuarios, 2</i>	
<i>proveedores, 1 anuncio con reporte (HTTP 200)</i>	<i>68</i>
<i>Figura 39</i>	<i>68</i>
<i>RF-21 — Admin visualiza anuncios reportados: anuncio de Carlos reportado por María</i>	
<i>(HTTP 200)</i>	<i>68</i>
<i>Figura 40</i>	<i>68</i>
<i>RF-18 — Admin lista proveedores activos filtrados por rol y estado (HTTP 200)</i>	<i>68</i>
<i>Figura 41</i>	<i>70</i>
<i>Swagger UI — Vista completa del grupo Administración con todos los endpoints y esquemas</i>	
<i>de datos</i>	<i>70</i>
<i>Figura 42</i>	<i>98</i>
<i>Figura 43</i>	<i>99</i>
<i>Figura 44</i>	<i>100</i>
<i>Figura 45</i>	<i>101</i>
<i>Figura 46</i>	<i>102</i>
<i>Figura 47</i>	<i>103</i>

Introducción

Actualmente, en Cuenca todavía se utilizan recomendaciones, grupos de mensajería y anuncios públicos para encontrar servicios independientes. Estos medios no siempre permiten verificar la información del proveedor antes de contratarlo.

A partir de esta situación, se desarrolló ServiLink, una plataforma web que permite publicar, buscar y contratar servicios independientes. La plataforma también incluye la verificación de proveedores y permite que sus usuarios ofrezcan servicios o los contraten.

El sistema utiliza Node.js, Express y MongoDB en el servidor, junto con React.js, Vite y Tailwind CSS en el cliente. Para organizar el desarrollo se utilizó Scrum y se realizaron pruebas funcionales y de integración antes de poner la plataforma en línea.

El trabajo está dividido en cuatro capítulos que abarcan el marco referencial, el marco teórico, el marco metodológico y los resultados de la investigación; este último capítulo integra, además, las conclusiones, recomendaciones y líneas de trabajo futuro.

Capítulo I

Marco referencial

1.1. Planteamiento del problema

La expansión de las tecnologías digitales y de los modelos de negocio basados en plataformas ha impulsado nuevas formas de intermediación entre quienes ofrecen y quienes requieren servicios. La OECD, la International Labour Organization y la European Union (2023) señalan que el trabajo mediado por plataformas digitales constituye una modalidad creciente de trabajo no estándar y que todavía existen vacíos de información para medir adecuadamente su alcance. A la vez, la International Labour Organization (2018) estimó que 2.000 millones de personas, equivalentes al 61,2 % de la población ocupada mundial, se encontraban en el empleo informal.

En Ecuador, el Plan de Creación de Oportunidades 2021-2025 incorporó entre sus metas el incremento del acceso a internet, mientras que la Agenda de Transformación Digital del Ecuador 2022-2025 planteó líneas de acción para fortalecer la apropiación de tecnologías digitales en distintos sectores (Secretaría Nacional de Planificación del Ecuador, 2021; Ministerio de Telecomunicaciones y de la Sociedad de la Información, 2022). Además, la Encuesta Nacional de Empleo, Desempleo y Subempleo anual permite analizar la situación del mercado laboral con desagregación provincial, lo que evidencia la importancia de abordar la informalidad mediante información oficial y soluciones adaptadas al contexto territorial (Instituto Nacional de Estadística y Censos [INEC], 2022).

En la ciudad de Cuenca, la contratación de servicios independientes —plomaría, electricidad, carpintería, servicios médicos, asesoría legal, entre otros— se realiza aún mediante referencias verbales, grupos de mensajería instantánea o letreros en espacios públicos, sin ningún mecanismo de verificación de identidad ni respaldo documental del prestador, lo que genera riesgos de seguridad y fraude para la ciudadanía que contrata. Esta situación evidencia la ausencia de una plataforma digital local que responda a estas necesidades de forma estructurada y segura, problemática que el presente proyecto busca atender mediante el desarrollo de ServiLink.

1.2. Justificación

El presente proyecto se justifica desde varias perspectivas complementarias. En el ámbito académico e institucional, la Ley Orgánica de Educación Superior (LOES) y el Reglamento de Régimen Académico del Consejo de Educación Superior (CES) establecen que los trabajos de titulación deben responder a problemáticas reales del entorno y generar soluciones con impacto social, lineamiento que la Universidad Católica de Cuenca promueve activamente en su modelo educativo, y que la Carrera de Software materializa a través de competencias en ingeniería de requerimientos, arquitectura de software y desarrollo de sistemas web, aplicadas directamente en este proyecto.

En el plano tecnológico, la propuesta integra un conjunto de herramientas modernas de desarrollo web —React 18, Vite, Tailwind CSS, Node.js, Express y MongoDB— seleccionadas según criterios de adecuación a los requerimientos, madurez, soporte de la comunidad y experiencia del equipo, ampliamente adoptadas en la construcción de plataformas

de intermediación de servicios, con una arquitectura cliente-servidor basada en API REST, autenticación mediante JSON Web Tokens (JWT) y gestión segura de archivos multimedia.

En el plano social, ServiLink responde directamente a la necesidad de formalizar la oferta de servicios independientes en Cuenca: al proveer un mecanismo de verificación de proveedores mediante revisión administrativa de documentos oficiales, permite a los ciudadanos contratar con garantías de seguridad, mientras que su esquema de doble rol — donde un mismo usuario puede actuar como cliente y como proveedor— amplía las oportunidades económicas de la población, en línea con los objetivos de la Agenda Digital Ecuador 2022-2025 del Ministerio de Telecomunicaciones.

Finalmente, en el ámbito económico, la digitalización de la oferta de servicios puede ampliar la visibilidad de los proveedores y facilitar a los clientes la búsqueda de alternativas disponibles. Los boletines estadísticos de ARCOTEL documentan la evolución del sector de telecomunicaciones y constituyen una referencia para contextualizar la infraestructura digital del país (Agencia de Regulación y Control de las Telecomunicaciones [ARCOTEL], 2023). No obstante, el presente trabajo no incluyó un levantamiento primario —mediante encuestas o entrevistas estructuradas— que cuantificara la frecuencia de contratación, la demanda específica o la percepción de riesgo en Cuenca; por ello, el posible efecto económico y social de ServiLink deberá evaluarse posteriormente con datos de adopción y uso.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar una plataforma web para la oferta y contratación de servicios independientes, mediante el análisis, diseño e implementación de sus módulos funcionales, con el fin de facilitar la conexión segura y verificada entre proveedores de servicios y ciudadanos que los requieren en la ciudad de Cuenca, Ecuador, 2026.

1.3.2. Objetivos específicos

- Realizar un análisis de los requerimientos funcionales y no funcionales necesarios para la plataforma web de oferta y contratación de servicios independientes, identificando las necesidades de los actores del sistema: clientes, proveedores y administradores.
- Diseñar la arquitectura del sistema, el modelo de base de datos y la interfaz de usuario, considerando criterios de escalabilidad, seguridad y orientación a la verificación de proveedores y la gestión de solicitudes de servicio.
- Implementar los módulos funcionales del sistema utilizando Node.js, Express, MongoDB y React.js, incluyendo autenticación por roles, verificación de proveedores, búsqueda de servicios y gestión de solicitudes con propuesta de citas.
- Validar el correcto funcionamiento del sistema mediante pruebas funcionales y de integración, y desplegarlo en un entorno de producción en la nube para su disponibilidad en línea.

1.4. Preguntas de investigación

Las preguntas de investigación del presente trabajo se derivan directamente de los objetivos específicos planteados y orientan el proceso metodológico adoptado para el desarrollo de ServiLink.

1.4.1. ¿De qué manera el análisis de requerimientos funcionales y no funcionales de los actores del sistema —clientes, proveedores y administradores— permite definir los módulos necesarios para una plataforma web de oferta y contratación de servicios independientes en Cuenca, Ecuador?

1.4.2. ¿Cómo debe diseñarse la arquitectura del sistema, el modelo de base de datos y la interfaz de usuario de una plataforma web de intermediación de servicios independientes para considerar criterios de escalabilidad, seguridad y soporte efectivo a los procesos de verificación de proveedores y gestión de solicitudes de servicio?

1.4.3. ¿De qué forma la implementación de los módulos funcionales de ServiLink utilizando Node.js, Express, MongoDB y React.js logra satisfacer los requerimientos de autenticación por roles, verificación de proveedores, búsqueda de servicios y gestión de solicitudes con propuesta de citas?

1.4.4. ¿En qué medida las pruebas funcionales y de integración aplicadas a ServiLink verifican el cumplimiento de los requerimientos establecidos y garantizan la disponibilidad operativa de la plataforma en un entorno de producción en la nube?

1.5. Alcance

El presente proyecto contempla el análisis, diseño e implementación de ServiLink, una plataforma web para la oferta y contratación de servicios independientes, aplicable a cualquier categoría profesional: oficios técnicos, profesionales de la salud, servicios de construcción y asesorías, entre otros. El sistema se entrega completamente funcional e incluye los siguientes módulos:

- Registro y autenticación de usuarios con roles diferenciados (cliente, proveedor, administrador), validación de cédula ecuatoriana para proveedores y verificación de cuenta por correo electrónico.
- Verificación de proveedores mediante revisión administrativa con estados de aprobación (pendiente, aprobado, rechazado) y notificación al proveedor sobre el resultado del proceso.
- Gestión de anuncios de servicios por parte de proveedores verificados, con soporte para carga de múltiples imágenes y visualización pública sin necesidad de autenticación.
- Búsqueda y filtrado de proveedores verificados por categoría de servicio, con acceso público al catálogo de anuncios.
- Gestión de solicitudes de servicio con propuesta de fecha y hora de cita, con flujo de aceptación, rechazo o propuesta de reprogramación por parte del proveedor.
- Panel de administración con gestión de usuarios, moderación de imágenes, edición de anuncios y atención de reportes sobre contenido inapropiado.

El sistema se desarrolla con Node.js y Express en el backend, MongoDB como base de datos, React.js con Vite y Tailwind CSS en el frontend, con despliegue en un entorno de producción en la nube. Quedan fuera del alcance del proyecto: el procesamiento de pagos en línea, el chat en tiempo real entre usuarios, la geolocalización con mapa interactivo, la aplicación móvil nativa y el sistema de calificaciones, funcionalidades que se identifican como trabajo futuro.

1.6. Metodología

El proyecto se desarrolla bajo la metodología ágil Scrum, ampliamente utilizada en proyectos de ingeniería de software por su adaptabilidad y su enfoque en entregas incrementales de valor. El trabajo se organiza en siete sprints de dos semanas, cada uno con objetivos, actividades y pruebas definidos, que en conjunto cubren cuatro etapas de trabajo: una etapa inicial de análisis y diseño, con el levantamiento de requerimientos, el diseño de la arquitectura y el modelado de la base de datos; una etapa de desarrollo del backend y la autenticación, que comprende la API REST y los módulos de registro, roles de usuario, verificación de proveedores, anuncios y solicitudes de servicio; una etapa de desarrollo del frontend y su integración con la API, junto con los dashboards de cliente, proveedor y administrador; y una etapa final de pruebas funcionales y de integración, corrección de errores y despliegue en un entorno de producción en la nube. El detalle de cada sprint se presenta en el Capítulo III.

Capítulo II

Marco teórico

2.1. Estado del arte

El presente trabajo se enmarca en la intersección de varios conceptos que conforman su título y que definen el alcance de la revisión de la literatura: plataforma web, oferta y contratación de servicios independientes, y verificación de proveedores. A continuación, se revisan primero estos conceptos desde la literatura especializada, y luego los trabajos académicos relacionados más relevantes del contexto ecuatoriano.

Plataforma web de intermediación de servicios

Una plataforma web de intermediación de servicios es un sistema de software accesible mediante navegador que actúa como canal entre dos o más grupos de usuarios —en este caso, proveedores y demandantes de servicios independientes— facilitando su conexión sin ser directamente prestadora del servicio. Parker, Van Alstyne y Choudary (2016) denominan a este modelo plataforma de dos lados (two-sided platform) y señalan que su valor reside en reducir los costos de búsqueda e información para ambas partes. A diferencia de las aplicaciones monolíticas tradicionales, las plataformas web modernas separan el frontend (interfaz de usuario, ejecutada en el navegador) del backend (lógica de negocio y persistencia de datos, ejecutada en el servidor), comunicados mediante una API REST. Esta arquitectura es la adoptada por ServiLink.

Oferta y contratación de servicios independientes

La contratación de servicios independientes hace referencia a la modalidad en que una persona ofrece sus habilidades o conocimientos de forma autónoma a terceros, sin relación de

dependencia laboral. A escala mundial, la International Labour Organization (2018) estimó que 2.000 millones de personas se encontraban en el empleo informal. En el ámbito digital, el trabajo mediado por plataformas comprende actividades productivas realizadas a través de una plataforma que organiza aspectos esenciales, como el acceso a clientes o la distribución del trabajo (OECD, International Labour Organization, & European Union, 2023). ServiLink se relaciona con este ámbito al facilitar la oferta y contratación digital de servicios independientes en el contexto local de Cuenca.

Verificación de proveedores en plataformas de servicios

La verificación de proveedores es el proceso mediante el cual una plataforma confirma que un prestador de servicios es quien dice ser, a través de la validación de documentos oficiales de identidad. Dentro de las plataformas de contratación, este control resulta determinante para dar confianza a los usuarios, disminuir las posibilidades de fraude y proteger a quien contrata el servicio (National Institute of Standards and Technology, 2017). Schwellnus et al. (2019) señalan que el principal obstáculo para la adopción de plataformas de servicios independientes es precisamente la confianza: los usuarios necesitan garantías razonables sobre la identidad del proveedor antes de contratar. ServiLink implementa este mecanismo mediante revisión administrativa de documentos, asignando a cada proveedor uno de tres estados —pendiente, aprobado, rechazado— que se refleja públicamente en su perfil.

Con estos conceptos como marco, a continuación se revisan los trabajos académicos relacionados más relevantes del contexto ecuatoriano, identificando los aportes que se retoman y los vacíos que ServiLink busca atender.

La revisión de antecedentes se desarrolló como una revisión narrativa orientada al diseño, debido a que en Ecuador la producción académica y las soluciones documentadas sobre plataformas locales de contratación de servicios independientes son limitadas. Se consultaron los repositorios institucionales de ESPOL, UPS, EPN y USFQ, empleando combinaciones de los términos “plataforma web”, “contratación de servicios”, “freelancers”, “servicios generales”, “intermediación” y “verificación de identidad”. Se incluyeron trabajos publicados entre 2018 y 2024 que describieran una solución tecnológica, su metodología, funcionalidades o validación. Para ampliar el contraste internacional se revisó además la documentación oficial de plataformas consolidadas de trabajo y servicios bajo demanda. Esta revisión no pretende demostrar la inexistencia absoluta de iniciativas en el país, sino establecer que, dentro de las fuentes académicas e institucionales localizables, no se identificó una solución local documentada que reúna todas las funciones definidas para ServiLink.

Espinoza y Barrera (2019) — Plataforma web para la contratación de freelancers de marketing digital

Espinoza y Barrera (2019), en la Escuela Superior Politécnica del Litoral, formularon un plan de negocio para una plataforma web que permite a profesionales independientes ofrecer servicios de marketing digital y a emprendedores acceder a ellos. El antecedente aporta una visión del modelo de intermediación y de la necesidad de aumentar la visibilidad de los trabajadores independientes. A diferencia de esa propuesta sectorial y orientada al análisis de negocio, ServiLink desarrolla una solución funcional para varias categorías de servicios e incorpora verificación documental de proveedores y gestión de solicitudes con citas.

Bajaña (2021) — Plataforma web basada en un modelo de intermediación de productos y servicios

Bajaña (2021), en la Universidad Politécnica Salesiana, diseñó una plataforma web de intermediación en la que el cliente publica una necesidad y los proveedores presentan ofertas. Este trabajo resulta relevante por la interacción estructurada entre demanda y oferta dentro de un portal web. ServiLink coincide en la función intermediadora, pero organiza la oferta mediante anuncios públicos de proveedores verificados y añade un flujo de contratación con aceptación, rechazo y propuesta de reagendamiento.

Alvear (2023) — Aplicación móvil para contratar profesionales de servicios generales

Alvear (2023), en la Escuela Politécnica Nacional, desarrolló una aplicación móvil multiplataforma para gestionar la contratación y solicitud de servicios generales. El proyecto utilizó Scrum, separó las aplicaciones de cliente y proveedor y evaluó su usabilidad mediante la escala SUS, con resultados favorables para ambos perfiles. Este antecedente confirma la viabilidad de digitalizar la contratación de servicios en Ecuador. ServiLink se diferencia por funcionar como plataforma web, permitir la acumulación de roles en una misma cuenta e incorporar verificación administrativa de identidad.

Romero Hernández (2023) — Plataforma web para la gestión de trabajos freelance en Ecuador

Romero Hernández (2023), en la Universidad San Francisco de Quito, desarrolló una plataforma web para gestionar trabajos freelance en Ecuador, enfocada en reducir costos asociados a plataformas internacionales, mejorar la comunicación y la gestión de tareas, y fortalecer la confianza mediante un mecanismo de protección financiera. Este trabajo comparte con ServiLink el contexto nacional y el propósito de conectar clientes con trabajadores

independientes. Sin embargo, ServiLink se concentra en servicios locales de distintas categorías, verifica documentalmente a los proveedores y estructura la coordinación mediante solicitudes y propuestas de cita, sin incluir pagos en línea.

En los antecedentes nacionales revisados se observa una tendencia hacia la digitalización de la oferta y contratación de servicios, aunque con alcances sectoriales, prototipos académicos o aplicaciones móviles. En el ámbito internacional, Upwork y Fiverr incorporan verificación de identidad para fortalecer la confianza en sus mercados digitales; Workana opera como plataforma de trabajo freelance con presencia latinoamericana; y Taskrabbbit vincula clientes con prestadores de tareas locales y contempla procesos de verificación y, según el mercado, comprobaciones de antecedentes. Estas plataformas demuestran la viabilidad del modelo de intermediación, pero responden a mercados, mecanismos de pago y condiciones regulatorias diferentes. ServiLink adapta el concepto al contexto de Cuenca mediante anuncios de servicios locales, verificación documental administrada, doble rol en una misma cuenta, propuesta y reagendamiento de citas y notificaciones por WhatsApp (Upwork, 2026; Fiverr, 2026; Workana, 2026; Taskrabbbit, 2026).

Comparación de antecedentes y plataformas de referencia

La Tabla 1 sintetiza las características relevantes de los trabajos nacionales y de las plataformas internacionales consultadas. La comparación se utiliza como evidencia documental para justificar las decisiones de diseño; no sustituye un estudio de mercado ni una encuesta de demanda en Cuenca.

Matriz comparativa de antecedentes nacionales y plataformas internacionales de referencia

Referencia	Ámbito	Tipo	Metodología/validación	Funciones relevantes	Limitación frente a ServiLink
Espinoza y Barrera (2019)	Ecuador	Plan de negocio web	Análisis de negocio	Oferta de freelancers de marketing digital	Sectorial; no implementa verificación documental ni citas
Bajaña (2021)	Ecuador	Diseño de plataforma	Diseño de intermediación	Publicación de necesidades y ofertas	No documenta doble rol, citas ni notificaciones
Herrera y Quishpe (2020)	Ecuador	Sistema web y móvil	Scrum	Catálogo y contratación de servicios del hogar	Tecnología y flujo distintos; no evidencia verificación documental administrativa
Alvear (2023)	Ecuador	Aplicación móvil	Scrum y evaluación SUS	Contratación de servicios generales	Aplicaciones separadas por rol; no es plataforma web
Romero Hernández (2023)	Ecuador	Plataforma web	Desarrollo de software	Gestión de trabajos freelance y protección financiera	Orientada a trabajo remoto y transacciones
Upwork / Fiverr	Internacional	Mercados freelance	Operación comercial	Perfiles, contratación y verificación de identidad	Orientadas principalmente a servicios digitales y pagos internacionales
Workana	Latinoamérica	Mercado freelance	Operación comercial	Proyectos, propuestas y contratación remota	No se orienta específicamente a oficinas y citas locales en Cuenca
Taskrabbitt	Internacional	Servicios bajo demanda	Operación comercial	Selección, agenda y tareas presenciales	Disponibilidad geográfica limitada y reglas por mercado
ServiLink	Cuenca, Ecuador	Plataforma web implementada	Scrum, pruebas funcionales e integración manual	Verificación documental, anuncios, doble rol, citas, WhatsApp y administración	Impacto y adopción real pendientes de un piloto longitudinal

2.2. Marco teórico referencial

Sistemas de intermediación de servicios

Los sistemas de intermediación de servicios son plataformas digitales que conectan a oferentes y demandantes, reduciendo los costos de búsqueda y la asimetría de información

propios de los mercados tradicionales. Parker, Van Alstyne y Choudary (2016) los describen como modelos de negocio en red que generan valor al facilitar intercambios entre dos o más grupos de usuarios interdependientes, incorporando mecanismos de confianza —verificación, calificación o garantías— que sustituyen las referencias personales propias del mercado informal. Schwellnus, Geva, Pak y Veiel (2019) señalan que el principal obstáculo para su adopción no es tecnológico sino la confianza: los usuarios necesitan garantías razonables sobre la identidad del proveedor antes de contratar.

Arquitectura cliente-servidor y APIs RESTful

La arquitectura cliente-servidor separa las funciones del sistema en un cliente que emite solicitudes y un servidor que las procesa y gestiona la persistencia (Tanenbaum & Van Steen, 2017). REST (Representational State Transfer) es el estilo arquitectónico para servicios web definido por Fielding (2000), que establece el uso de métodos HTTP estándar, ausencia de estado en el servidor y recursos identificados por URI. ServiLink implementa esta arquitectura con React 18 como cliente y Node.js/Express como servidor, comunicados mediante HTTPS/REST con autenticación JWT.

Autenticación con JSON Web Tokens (JWT)

JWT es un estándar abierto definido en el RFC 7519 (Jones, Bradley, & Sakimura, 2015) que especifica un formato compacto para transmitir información de forma segura entre partes mediante un objeto JSON firmado digitalmente, compuesto por encabezado, carga útil y firma. En ServiLink, el JWT incluye el identificador del usuario y su arreglo de roles[], lo que permite que el servidor tome decisiones de autorización sin consultar la base de datos en cada solicitud.

Control de acceso basado en roles (RBAC)

El modelo RBAC asigna permisos a roles y los roles a usuarios, en lugar de asignar permisos directamente a cada usuario (Sandhu, Ferraiolo, & Kuhn, 2000). ServiLink adopta este modelo mediante un arreglo `roles[]` en el documento de usuario de MongoDB, que permite la acumulación de múltiples roles —cliente y proveedor— en una sola cuenta, habilitando el modelo de doble rol sin duplicación de cuentas.

Hashing de contraseñas con bcrypt

Bcrypt es una función de hashing para contraseñas que incorpora un factor de costo configurable y una sal aleatoria por cada contraseña, dificultando ataques de fuerza bruta. El Open Web Application Security Project (OWASP, 2024) lo reconoce como uno de los algoritmos aceptables para almacenamiento de contraseñas con un factor de costo mínimo de 10.

Verificación de identidad digital

El National Institute of Standards and Technology (2017) distingue entre autenticación —verificar que el usuario es quien dice ser en una sesión— y verificación de identidad —confirmar que la identidad detrás de la cuenta es real y legítima—. ServiLink incorpora ambos niveles: autenticación mediante JWT y bcrypt, y verificación de identidad mediante validación del dígito verificador módulo 10 de la cédula ecuatoriana y revisión administrativa de documentos oficiales.

Node.js y Express

Node.js es un entorno de ejecución de JavaScript del lado del servidor con modelo de E/S no bloqueante y dirigido por eventos, eficiente para aplicaciones con múltiples conexiones concurrentes (OpenJS Foundation, 2024c). Sobre Node.js se emplea Express, una capa ligera que aporta el sistema de rutas y los middlewares necesarios para levantar servicios REST (OpenJS Foundation, 2024a). Multer es el middleware para Express que procesa datos multipart/form-data, utilizado en ServiLink para la carga de imágenes y documentos (OpenJS Foundation, 2024b).

React, Vite y Tailwind CSS

En el cliente se utiliza React, la biblioteca de Meta que organiza la interfaz en componentes reutilizables y actualiza la vista de forma eficiente mediante un DOM virtual (Meta Platforms, 2023). Vite provee un servidor de desarrollo con recarga en caliente instantánea mediante módulos ECMAScript nativos (Vite, 2024). Para los estilos se emplea Tailwind CSS, cuyo enfoque de clases utilitarias permite construir interfaces responsivas combinando clases predefinidas directamente en el marcado HTML (Tailwind Labs, 2024).

MongoDB y Mongoose

MongoDB es un sistema de gestión de bases de datos NoSQL orientado a documentos BSON, con esquemas flexibles que facilitan estructuras de datos variables como el arreglo de roles por usuario (MongoDB Inc., 2024). Mongoose es el ODM para MongoDB en Node.js que añade esquemas, validaciones y middlewares sobre la flexibilidad nativa de MongoDB (Mongoose, 2024).

WhatsApp Business Cloud API y estándar E.164

La WhatsApp Business Cloud API de Meta permite enviar mensajes de WhatsApp de forma programática mediante HTTPS/REST sin infraestructura propia (Meta for Developers, 2024). El estándar E.164 de la ITU-T define el formato internacional de numeración telefónica con prefijo de país precedido de «+» (International Telecommunication Union — Telecommunication Standardization Sector [ITU-T], 2018). ServiLink normaliza los números locales ecuatorianos (0XXXXXXXXXX) al formato E.164 (+593XXXXXXXXXX) antes de cada envío.

Railway y despliegue en la nube

El alojamiento se resuelve con Railway, un servicio en la nube que agiliza la publicación de aplicaciones web al conectarse directamente con repositorios Git, emitir certificados TLS de forma automática y centralizar las variables de entorno (Railway, 2024). ServiLink se despliega íntegramente en Railway con pipeline continuo desde la rama main del repositorio.

Scrum como metodología de desarrollo

Scrum es un marco de trabajo ágil basado en ciclos iterativos breves denominados sprints, que permiten entregas incrementales de valor y adaptación continua a los requerimientos. Sus artefactos principales son el Product Backlog —lista priorizada de funcionalidades—, el Sprint Planning —planificación de cada iteración—, y la Sprint Review y Retrospective —revisión del producto y del proceso al cierre de cada ciclo (Schwaber & Sutherland, 2020).

Capítulo III

Marco metodológico

3.1. Enfoque metodológico

El presente trabajo adopta un enfoque de investigación aplicada con carácter tecnológico, orientado al diseño e implementación de una solución de software funcional que responde a una problemática identificada en el contexto real de la ciudad de Cuenca. Este tipo de enfoque se seleccionó porque el objetivo no es generar conocimiento teórico nuevo, sino producir un artefacto —una plataforma web— que resuelva un problema concreto y verificable mediante pruebas empíricas sobre el sistema resultante (Hernández Sampieri, Fernández Collado, & Baptista Lucio, 2014).

La naturaleza del problema exigió combinar dos modalidades de investigación complementarias. La investigación documental sustentó la fase de análisis: se revisó literatura sobre plataformas de intermediación de servicios, modelos de verificación de identidad digital y trabajos relacionados del contexto ecuatoriano para fundamentar las decisiones de diseño. El desarrollo experimental articuló las fases subsiguientes: el sistema se construyó de forma iterativa, verificando en cada ciclo que los módulos implementados cumplieran los requerimientos establecidos antes de avanzar al siguiente.

Optar por este enfoque implicó asumir que el proceso de desarrollo mismo genera conocimiento: las decisiones técnicas tomadas durante la implementación —cómo estructurar el modelo de doble rol, cómo gestionar la normalización de números telefónicos para WhatsApp, cómo resolver la autorización granular por propietario de recurso en las solicitudes

de servicio— no estaban completamente definidas en la etapa de diseño, sino que se fueron documentando y refinando conforme avanzaba el proyecto. Esta característica es propia de los proyectos de ingeniería de software de mediana complejidad y justifica el uso de una metodología ágil iterativa.

3.2. Flujo de trabajo

El desarrollo de ServiLink se organizó bajo el marco de trabajo Scrum, adaptado al contexto de un proyecto individual de titulación. Scrum estructura el trabajo en sprints — iteraciones cortas y delimitadas— que permiten validar avances parciales con la directora del proyecto antes de continuar (Schwaber & Sutherland, 2020). Cada sprint parte de un Sprint Planning donde se seleccionan los ítems del Product Backlog a desarrollar, y cierra con una revisión del incremento entregable.

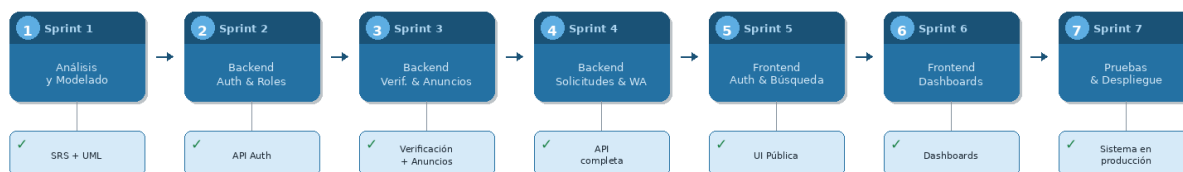
El proceso de análisis comenzó en marzo de 2026. Durante ese mes se definieron el problema, los actores y los flujos principales mediante revisión documental de trabajos relacionados y análisis de plataformas de intermediación. Entre marzo y abril se elaboró la primera versión del SRS y se realizaron sesiones de revisión con la directora del proyecto; las observaciones se incorporaron de manera iterativa hasta consolidar 21 requerimientos funcionales y 19 no funcionales. La priorización se realizó según criticidad para el flujo principal, dependencia técnica y valor para los actores: autenticación y verificación, contratación y citas, administración y despliegue. Desde abril hasta junio, los requerimientos se transformaron en ítems del Product Backlog y se implementaron en siete sprints de dos semanas. La validación del SRS se efectuó mediante la revisión de coherencia entre objetivos, actores, casos de uso, diagramas, módulos implementados y casos de prueba. Debido al

carácter tecnológico del proyecto y a la escasez de estudios locales específicos, la elicitación se sustentó principalmente en evidencia documental y revisión técnica, no en encuestas de mercado.

La Figura 1 presenta el plan de sprints del proyecto con los entregables asociados a cada iteración.

Figura 1

Plan de sprints del proyecto ServiLink — 7 iteraciones de dos semanas con entregables verificables.



3.3. Análisis de requerimientos

3.3.1. Actores del sistema

El levantamiento de requerimientos se efectuó mediante análisis documental del problema, revisión de los antecedentes descritos en el Capítulo II, definición de los flujos operativos esperados y revisión iterativa con la directora del proyecto. No se aplicaron encuestas ni entrevistas estructuradas a una muestra externa durante esta fase. A partir de este procedimiento se identificaron tres actores principales cuyas necesidades guiaron la definición de los requerimientos funcionales:

- Cliente: usuario que busca y contrata servicios independientes. Puede enviar solicitudes de servicio con propuesta de cita, hacer seguimiento y reportar anuncios inapropiados.
- Proveedor: profesional o técnico independiente que ofrece servicios en la plataforma. Solo los proveedores cuya identidad ha sido verificada administrativamente pueden publicar anuncios y gestionar solicitudes.
- Administrador: responsable de la gestión del sistema. Revisa verificaciones, modera anuncios reportados y gestiona cuentas de usuario.

El sistema contempla además el actor dual: el usuario con ambos roles activos simultáneamente, que puede actuar como cliente o como proveedor desde una única cuenta, con la restricción de que no puede enviarse solicitudes de servicio a sí mismo (RF-10).

3.3.2. Requerimientos funcionales

El documento de especificación de requerimientos elaborado para el presente proyecto identifica 21 requerimientos funcionales agrupados en seis módulos. La Tabla 1 presenta la matriz consolidada.

Tabla 1

Matriz de requerimientos funcionales de ServiLink

ID	Nombre	Módulo	Actor	Prioridad
RF-01	Registro de cliente	Autenticación	Cliente	Alta
RF-02	Registro de proveedor con validación de cédula	Autenticación	Proveedor	Alta

ID	Nombre	Módulo	Actor	Prioridad
RF-03	Inicio de sesión con JWT	Autenticación	Todos	Alta
RF-04	Cierre de sesión	Autenticación	Todos	Media
RF-05	Carga de documentos de identidad	Verificación	Proveedor	Alta
RF-06	Revisión administrativa de verificación	Verificación	Administrador	Alta
RF-07	Notificación WhatsApp: resultado de verificación	Verificación	Sistema	Alta
RF-08	Búsqueda y filtrado de proveedores	Búsqueda	Cliente/Dual	Alta
RF-09	Perfil público del proveedor verificado	Búsqueda	Cliente/Dual	Alta
RF-10	Envío de solicitud de servicio con cita	Solicitudes	Cliente/Dual	Alta
RF-11	Gestión de solicitudes por el proveedor	Solicitudes	Proveedor	Alta
RF-12	Notificación WhatsApp: eventos de cita	Solicitudes	Sistema	Alta
RF-13	Seguimiento de solicitudes por el cliente	Solicitudes	Cliente/Dual	Alta
RF-14	Proveedor actuando como cliente (rol dual)	Rol dual	Proveedor	Alta
RF-15	Upgrade de rol: cliente → proveedor	Rol dual	Cliente	Alta
RF-16	Revisión de upgrade por el administrador	Rol dual	Administrador	Alta
RF-17	Panel de administración centralizado	Administración	Administrador	Media
RF-18	Gestión de usuarios	Administración	Administrador	Media
RF-19	Publicación y gestión de anuncios	Anuncios	Proveedor	Alta
RF-20	Reporte de anuncios inapropiados	Anuncios	Todos (autent.)	Alta
RF-21	Gestión de anuncios reportados	Anuncios	Administrador	Alta

3.3.3. Requerimientos no funcionales

El mismo documento define 19 requerimientos no funcionales. La Tabla 2 presenta los de mayor impacto en las decisiones de diseño e implementación.

Tabla 2*Requerimientos no funcionales principales de ServiLink*

ID	Categoría	Descripción	Prioridad
RNF-01	Rendimiento	El tiempo de respuesta de la API para operaciones de lectura (búsqueda, listado) no debe superar los 2 segundos bajo condiciones normales.	Alta
RNF-02	Rendimiento	El sistema debe soportar al menos 50 usuarios concurrentes sin degradación perceptible del rendimiento.	Media
RNF-03	Rendimiento	Las imágenes de perfil y documentos cargados deben procesarse y almacenarse en menos de 5 segundos.	Media
RNF-04	Seguridad	Toda comunicación entre cliente y servidor debe realizarse sobre HTTPS con certificado TLS válido.	Alta
RNF-05	Seguridad	Las contraseñas se almacenarán con hash bcrypt, con factor de coste mínimo de 10 rondas; nunca en texto plano.	Alta
RNF-06	Seguridad	Los endpoints de la API estarán protegidos mediante JWT; los de administración verificarán adicionalmente el rol de administrador.	Alta
RNF-07	Seguridad	El sistema implementará protección contra inyección NoSQL en todas las consultas a MongoDB.	Alta
RNF-08	Seguridad	Los documentos de identidad se almacenarán en una ruta no pública, accesible solo por el administrador autenticado.	Alta
RNF-09	Seguridad	El token de acceso de WhatsApp Business API se almacenará como variable de entorno y nunca se expondrá en el repositorio.	Alta
RNF-10	Usabilidad	La interfaz debe ser responsiva y funcionar en dispositivos móviles desde 360 px de ancho.	Alta
RNF-11	Usabilidad	Los formularios mostrarán mensajes de error claros y descriptivos junto al campo correspondiente.	Media
RNF-12	Usabilidad	El flujo de registro no debe requerir más de 3 pasos para completarse.	Media
RNF-13	Usabilidad	La interfaz estará completamente en español, adaptada al contexto ecuatoriano.	Alta
RNF-14	Usabilidad	El dashboard del usuario con rol dual mostrará claramente las secciones "Como cliente" y "Como proveedor", diferenciadas visualmente.	Alta
RNF-15	Mantenibilidad	El código fuente estará organizado por responsabilidad (controllers, routes, models, middlewares, services) siguiendo el patrón MVC.	Media

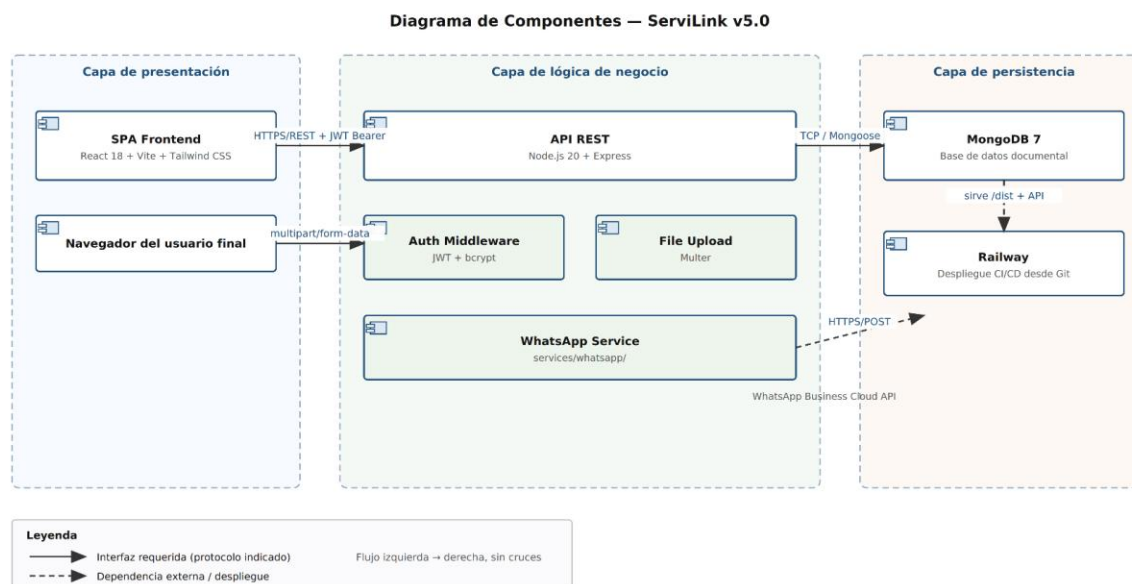
RNF-16	Mantenibilidad	Las variables sensibles (base de datos, JWT secret, token y Phone Number ID de WhatsApp) se gestionarán con un archivo .env y nunca se subirán al repositorio.	Alta
RNF-17	Escalabilidad	La arquitectura permitirá agregar nuevas categorías de servicio y nuevos eventos de notificación WhatsApp sin modificar la lógica central.	Media
RNF-18	Disponibilidad	El sistema debe estar disponible en línea con una disponibilidad objetivo del 99 % mensual.	Media
RNF-19	Despliegue	El sistema se desplegará en Railway con un pipeline de despliegue continuo desde Git.	Alta

3.4. Diseño de la arquitectura

ServiLink implementa una arquitectura cliente-servidor en tres capas que separa con claridad las responsabilidades de presentación, lógica de negocio y persistencia. La Figura 2 presenta el diagrama de componentes del sistema con las interfaces de comunicación entre cada capa.

Figura 2

Diagrama de componentes



La Figura 2 se organiza de izquierda a derecha en tres capas. En la **capa de presentación** se ubican el navegador del usuario final y la SPA Frontend construida con React 18, Vite y Tailwind CSS, responsable de renderizar las vistas y gestionar el estado de sesión. La **capa de lógica de negocio** contiene la API REST en Node.js 20 con Express y tres componentes internos: el Auth Middleware, que valida el token JWT y aplica un hash a las contraseñas con bcrypt; el File Upload basado en Multer, que recibe los documentos de identidad y las imágenes de anuncios; y el WhatsApp Service, encapsulado en services/whatsapp/, que construye y envía las plantillas de notificación. La **capa de persistencia** agrupa la base de datos documental MongoDB 7 y el entorno de despliegue Railway, que sirve tanto el directorio /dist del frontend como la API mediante integración continua desde Git.

Las interfaces entre capas están etiquetadas con su protocolo. La SPA se comunica con la API mediante **HTTPS/REST**, enviando el token JWT en el encabezado Authorization Bearer; la carga de archivos emplea **multipart/form-data**; la API accede a la base de datos por **TCP mediante Mongoose**; y el WhatsApp Service consume la WhatsApp Business Cloud API por **HTTPS/POST** como dependencia externa. El flujo no presenta cruces entre conectores, lo que evidencia la separación estricta de responsabilidades entre capas.

Capa de presentación. SPA desarrollada en React 18 con Vite y Tailwind CSS, que se comunica con el backend mediante HTTPS/REST con el token JWT en el encabezado Authorization Bearer. El directorio de producción (/dist) se sirve desde el mismo servicio Express de Railway, usando enrutamiento relativo (VITE_API_URL vacía) para evitar problemas de CORS en producción.

Capa de lógica de negocio. API REST en Node.js 20 con Express, organizada en controllers/, routes/, models/, middlewares/ y services/whatsapp/. La ruta GET /api/health actúa como health check en Railway. Las variables sensibles se gestionan con dotenv.

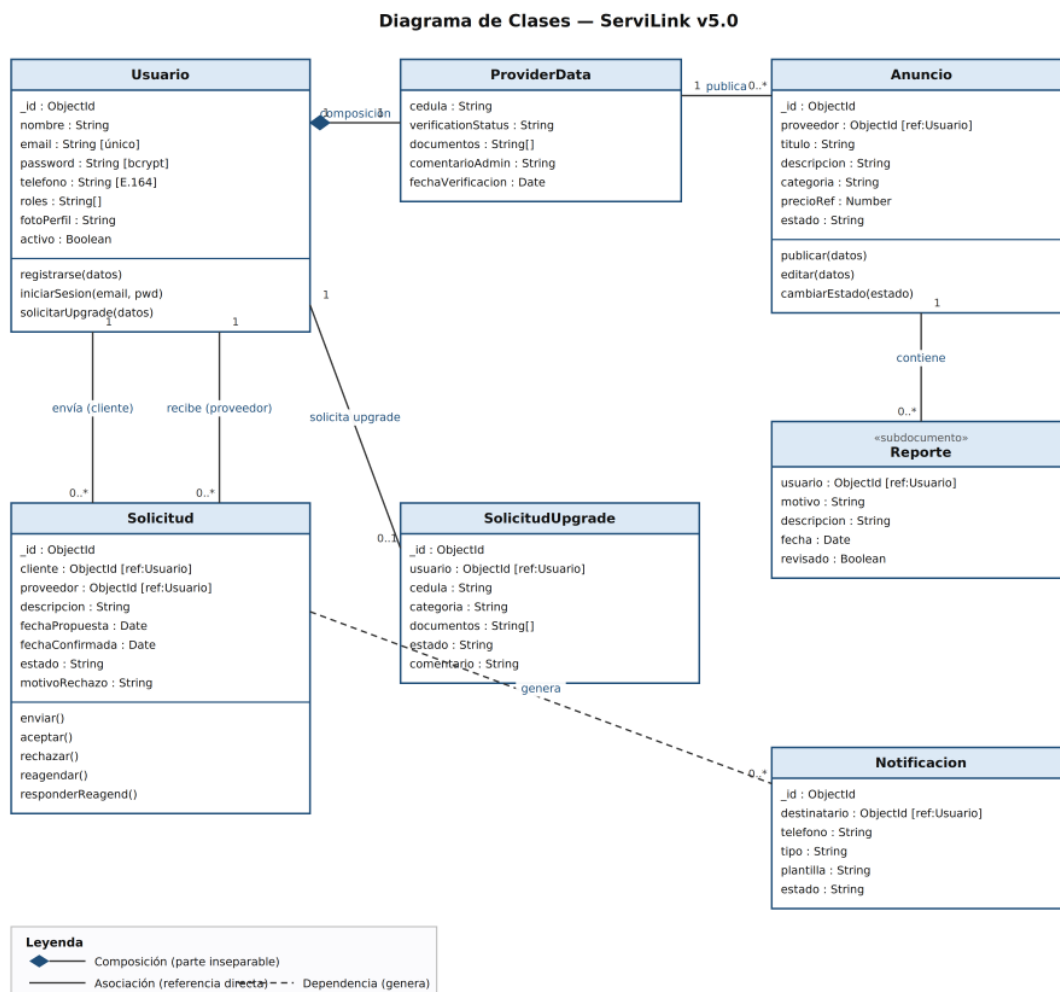
Capa de persistencia. MongoDB 7 en Railway, accedida mediante Mongoose. La cadena de conexión incluye /servilink?authSource=admin para autenticación correcta en producción.

3.5. Diseño de la base de datos

El modelo de datos se estructura en cinco colecciones principales en MongoDB. La Figura 3 presenta el diagrama de clases con las entidades, atributos y relaciones del sistema.

Figura 3

Diagrama de clases



El diagrama de la Figura 3 comprende siete entidades. **Usuario** es la entidad central y almacena nombre, correo único, contraseña protegida mediante hash con bcrypt, teléfono en formato E.164, el arreglo roles[] que habilita el rol dual, la fotografía de perfil y el estado de la cuenta; expone los métodos registrarse(), iniciarSesion() y solicitarUpgrade(). **ProviderData** se vincula a Usuario mediante una relación de composición y concentra los datos de verificación del proveedor: cédula, verificationStatus, documentos cargados, comentario del administrador y fecha de resolución. **Anuncio** registra el servicio publicado

con título, descripción, categoría, precio referencial y estado, junto con los métodos publicar(), editar() y cambiarEstado(); mantiene una referencia al proveedor que lo creó.

Reporte es un subdocumento contenido en Anuncio que guarda el usuario que reporta, el motivo, la descripción, la fecha y el indicador de revisión. **Solicitud** modela la contratación del servicio y referencia tanto al cliente como al proveedor, con la descripción, la fecha propuesta, la fecha confirmada, el estado y el motivo de rechazo; sus métodos enviar(), aceptar(), rechazar(), reagendar() y responderReagend() cubren el ciclo completo de la cita. **SolicitudUpgrade** representa la petición de ascenso de cliente a proveedor e incluye cédula, categoría, documentos, estado y comentario administrativo. Finalmente, **Notificacion** almacena el destinatario, el teléfono, el tipo, la plantilla utilizada y el estado de envío de cada mensaje de WhatsApp.

En cuanto a las relaciones, Usuario mantiene una composición con ProviderData y asociaciones de uno a muchos con Anuncio (publica), con Solicitud —diferenciada en **envía (cliente)** y **recibe (proveedor)**— y una relación de cero a uno con SolicitudUpgrade. Anuncio contiene cero o más instancias de Reporte, y Solicitud genera notificaciones mediante una relación de dependencia.

La Tabla 3 describe las colecciones con sus campos y notas de diseño más relevantes.

Tabla 3*Colecciones principales de la base de datos de ServiLink*

Colección	Campos principales	Notas de diseño
users	nombre, email, password (hash bcrypt), roles[] (arreglo), cédula, teléfono, estado, providerData	El arreglo roles[] permite acumulación de roles sin duplicar cuentas (RBAC). La cédula se valida con el algoritmo módulo 10.
announcements	título, descripción, categoría, precio, imágenes[], estado (activo/inactivo/suspendido), proveedor (ref.)	Solo anuncios con estado 'activo' aparecen en búsquedas. Múltiples imágenes por anuncio.
service_requests	descripción, fecha/hora propuesta, estado, cliente (ref.), proveedor (ref.), historial reagendamientos[]	El historial registra cada propuesta de nueva fecha. Referencia cruzada a users para ambos actores.
upgrade_requests	cliente (ref.), documentos[], estado (pendiente/aprobado/rechazado), comentario admin, fecha resolución	Al aprobarse, roles[] del usuario se actualiza añadiendo 'proveedor'.
reports	anuncio (ref.), reportante (ref.), motivo, estado (pendiente/revisado)	Un usuario no puede reportar el mismo anuncio más de una vez.

3.6. Casos de uso — descripción narrativa

El SRS documenta cuatro casos de uso narrativos principales, seleccionados por representar los flujos de mayor interacción y riesgo funcional: registro y autenticación, verificación de proveedores, gestión de anuncios y contratación de servicios. Estos casos no equivalen a cuatro requerimientos aislados, sino que agrupan operaciones relacionadas. La cobertura del conjunto de los 21 requerimientos funcionales se complementa con los ocho diagramas de casos de uso y la matriz de trazabilidad presentada en los anexos.

3.6.1. CU-01 — Contratar un servicio

Actor principal: cliente o usuario con rol dual. Actores secundarios: proveedor destino y sistema de notificaciones. Precondición: el solicitante está autenticado y existe al menos un proveedor verificado en la categoría buscada. Flujo principal: el usuario accede al buscador; ingresa palabra clave o categoría; el sistema muestra proveedores verificados; el usuario selecciona uno y visualiza su perfil; pulsa 'Solicitar servicio' y completa descripción, fecha y hora propuesta; el sistema registra la solicitud con estado 'pendiente' y envía notificación WhatsApp al proveedor. Flujo alternativo: si el solicitante es el mismo que el proveedor destino (rol dual), el sistema bloquea la acción (RF-10). Postcondición: solicitud activa en estado 'pendiente' y proveedor notificado.

3.6.2. CU-02 — Gestionar una solicitud recibida

Actor principal: proveedor. Actores secundarios: cliente y sistema de notificaciones. Precondición: proveedor autenticado con al menos una solicitud 'pendiente'. El caso de uso contempla tres flujos: confirmación (estado → 'aceptada', WhatsApp al cliente), rechazo (estado → 'rechazada' con motivo opcional, WhatsApp al cliente) y reagendamiento (estado → 'reagendada' con nueva propuesta, WhatsApp al cliente). Postcondición: solicitud con estado definitivo y ambas partes notificadas.

3.6.3. CU-03 — Solicitar upgrade de rol

Actor principal: cliente. Actores secundarios: administrador y sistema. Precondición: usuario con solo rol 'cliente' sin upgrade pendiente. Flujo: el cliente accede a su perfil, selecciona 'Quiero ser proveedor', completa el formulario con cédula y datos profesionales, carga documentos de identidad; el sistema valida la cédula con el algoritmo módulo 10; registra

la solicitud como 'pendiente'; el administrador revisa y aprueba o rechaza; el sistema notifica al cliente vía WhatsApp. Al aprobarse, roles[] se actualiza a ['cliente', 'proveedor']. Flujo alternativo: cédula ya registrada en otro perfil → error. Postcondición: usuario con rol dual o conserva solo 'cliente'.

3.6.4. CU-04 — Verificar un proveedor

Actor principal: administrador. Actores secundarios: proveedor y sistema. Precondición: administrador autenticado con solicitudes de verificación 'pendientes'. Flujo: el administrador accede al panel de verificaciones; selecciona un proveedor; visualiza los documentos cargados; selecciona 'Aprobar' o 'Rechazar' e ingresa comentario; el sistema actualiza el estado y envía WhatsApp al proveedor con el resultado y, en caso de rechazo, el motivo. Postcondición: proveedor con estado 'aprobado' o 'rechazado' y notificado.

3.7. Descripción de los diagramas UML

El modelado UML de ServiLink se documenta mediante 20 figuras organizadas en cuatro tipos de diagrama que representan perspectivas complementarias del sistema.

3.7.1. Diagramas de casos de uso

Los ocho diagramas de casos de uso organizan los 21 RF en módulos cohesivos. El diagrama de autenticación (Figura 4) cubre RF-01 a RF-04. El de rol dual (Figura 5) cubre RF-14, RF-15 y RF-16 y representa mediante asociaciones directas las interacciones de solicitud y revisión del cambio de rol. Los diagramas de búsqueda y perfil (Figura 6), solicitudes y citas (Figura 7), gestión de anuncios (Figura 8), verificación (Figura 9), panel de administración (Figura 10) y notificaciones WhatsApp (Figura 11) completan la representación funcional.

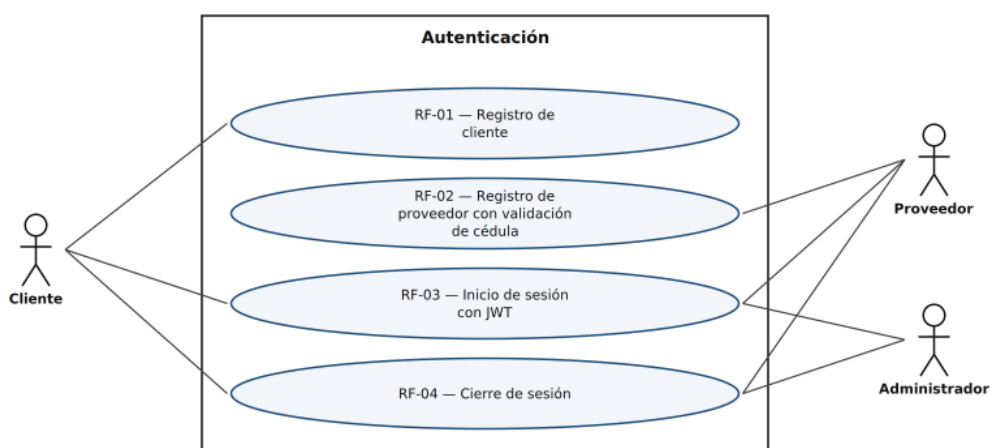
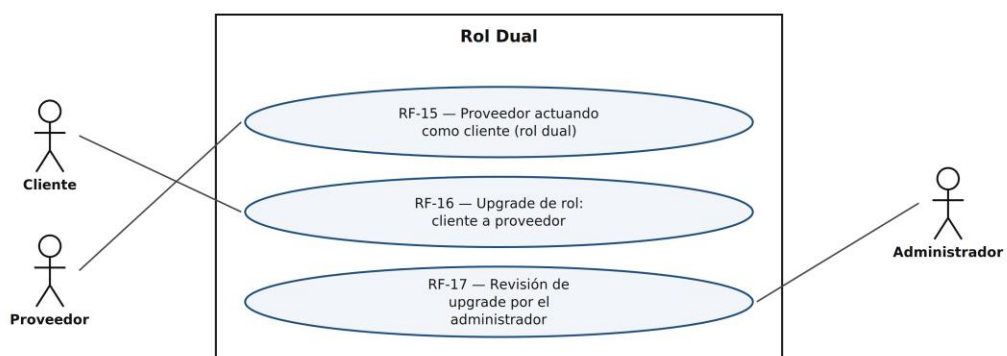
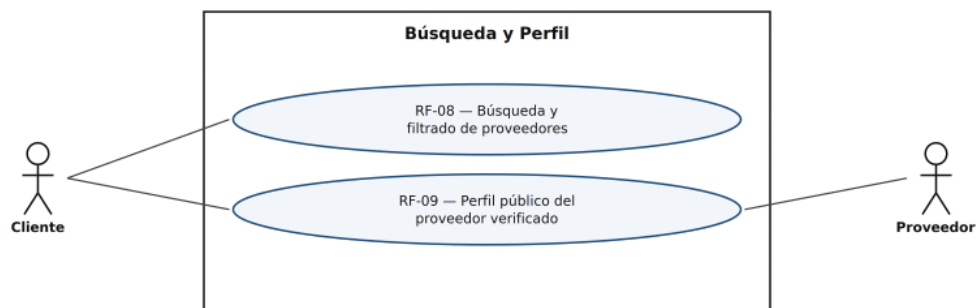
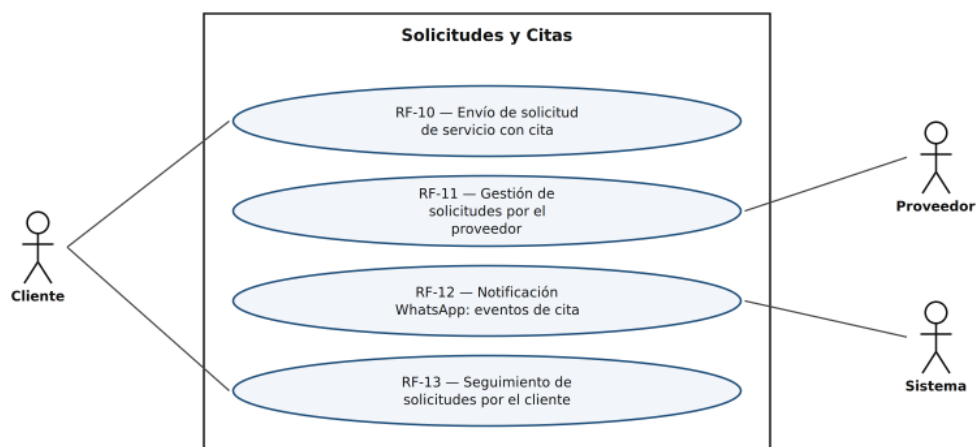
Figura 4*Casos de uso — Autenticación***Figura 5***Casos de uso — Rol dual*

Figura 6

Casos de uso — Búsqueda y perfil

**Figura 7**

Casos de uso — Solicitudes y citas

**Figura 8**

Casos de uso — Gestión de anuncios

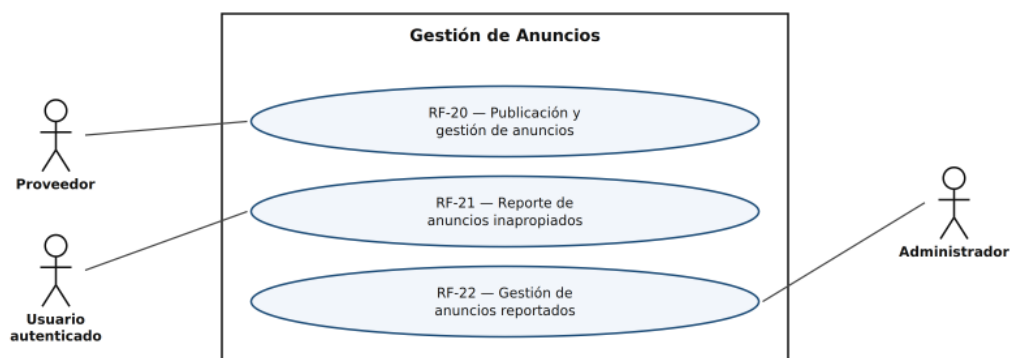


Figura 9

Casos de uso — Verificación de proveedores

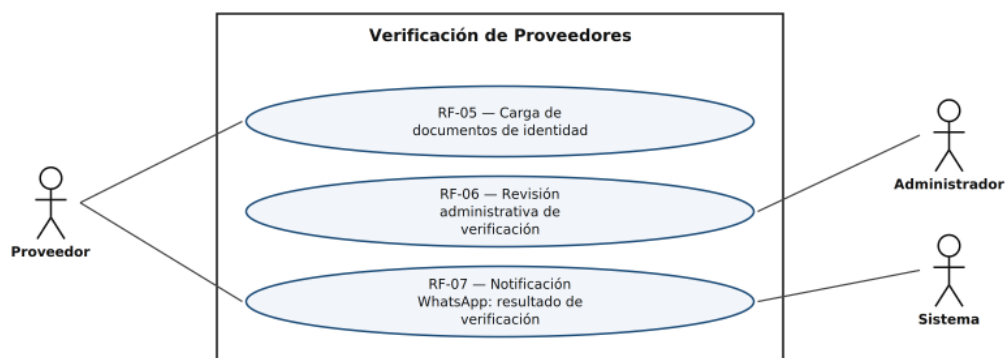


Figura 10

Casos de uso — Panel de administración

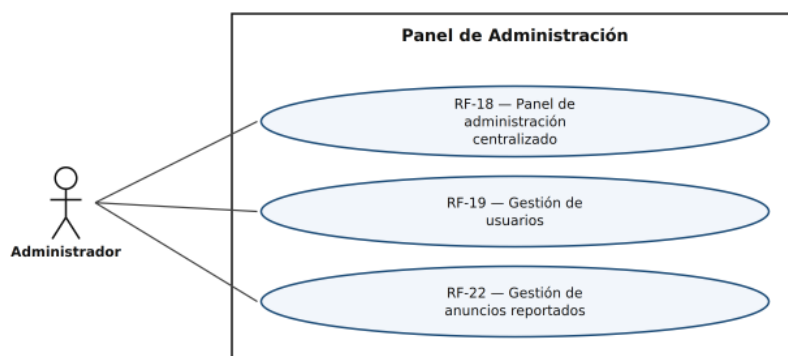
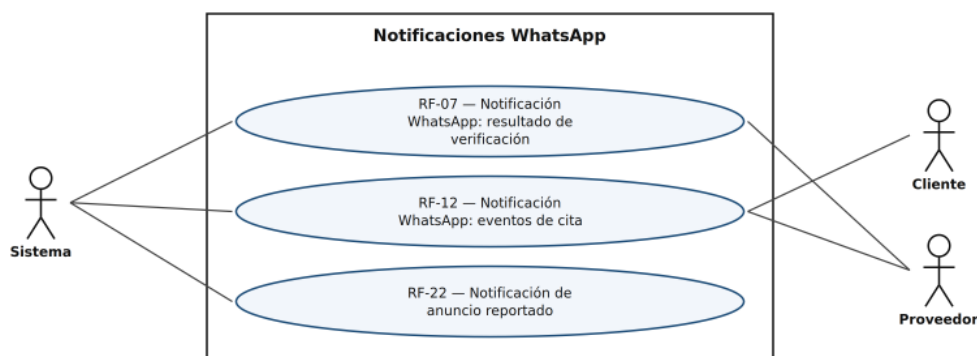


Figura 11

Casos de uso — Notificaciones WhatsApp



3.7.2. Diagramas de secuencia

Los diez diagramas de secuencia documentan los flujos temporales entre participantes. Las flechas sólidas son mensajes síncronos y las punteadas son respuestas; los marcos 'alt' representan flujos alternativos. Se presentan los cuatro más representativos: registro de proveedor con validación de cédula (Figura 12), contratación completa de servicio (Figura 13), verificación de proveedor (Figura 14) y upgrade de rol (Figura 15); los seis diagramas de secuencia restantes se presentan en el Anexo C.

Figura 12

Diagrama de secuencia — Registro de proveedor: validación de cédula módulo 10, Multer, cuenta con verificationStatus pendiente

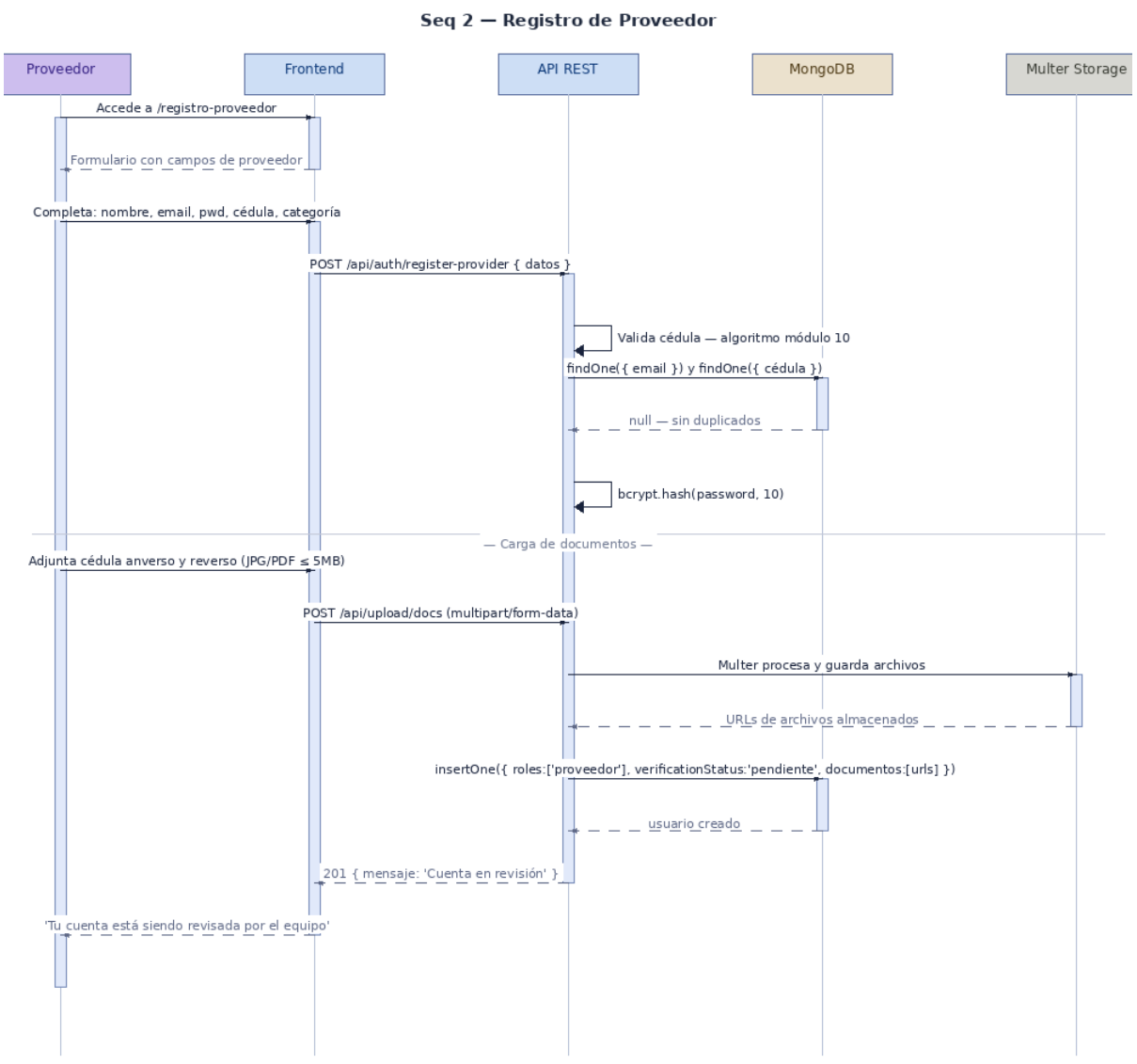


Figura 13

Diagrama de secuencia — Contratación completa de servicio: búsqueda, solicitud, notificación WhatsApp y aceptación

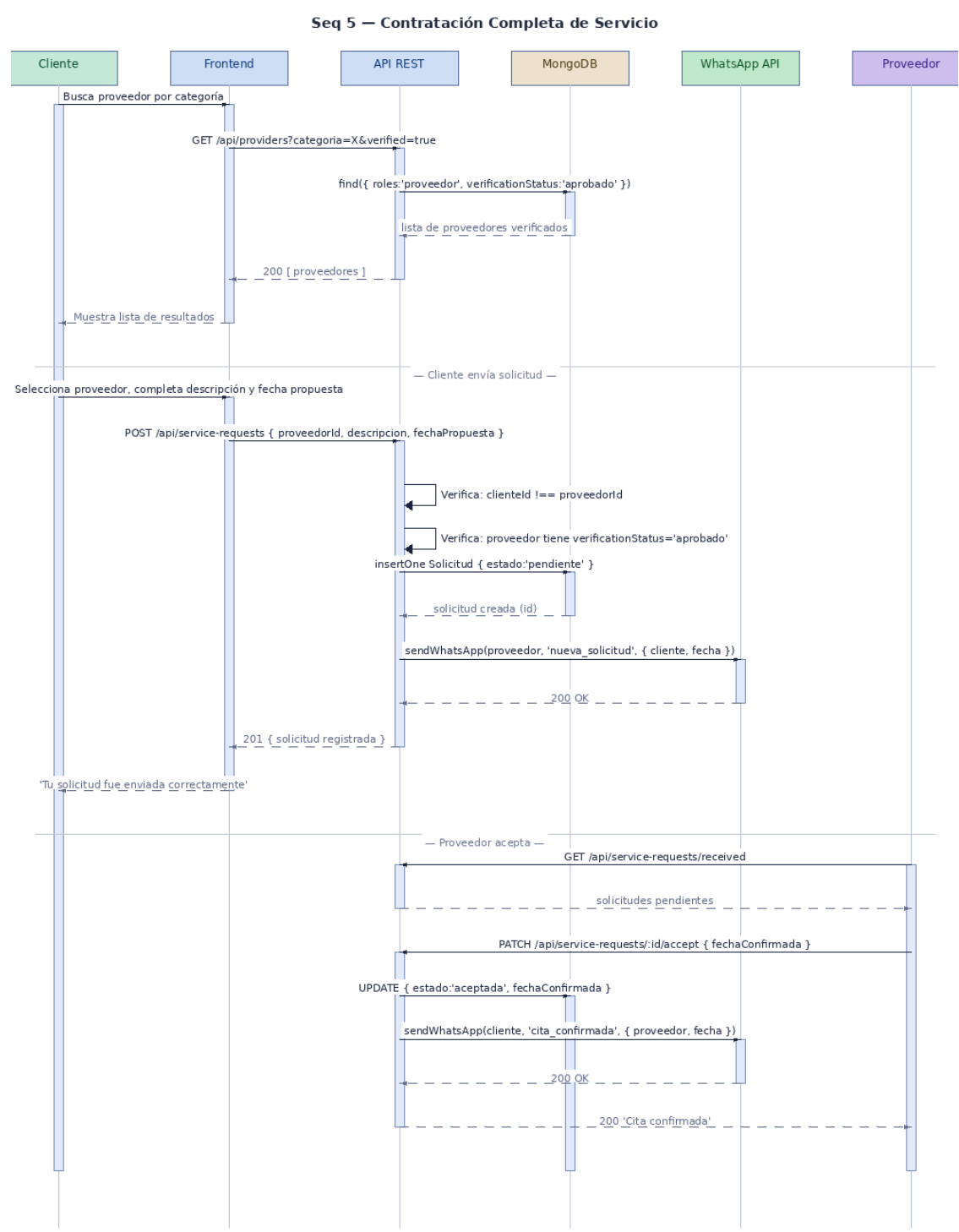


Figura 14

Diagrama de secuencia — Verificación de proveedor: aprobación y rechazo con notificación WhatsApp

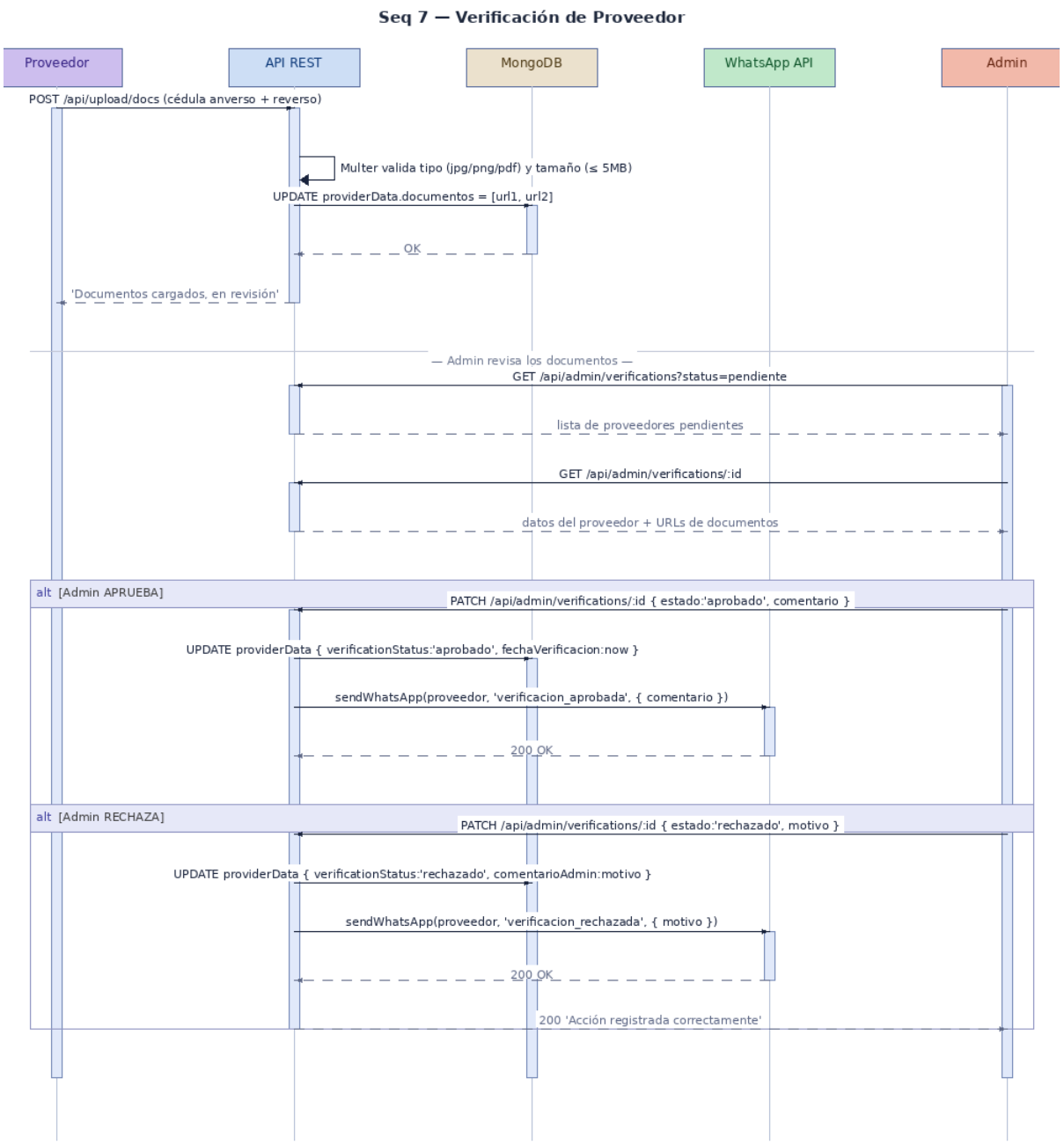
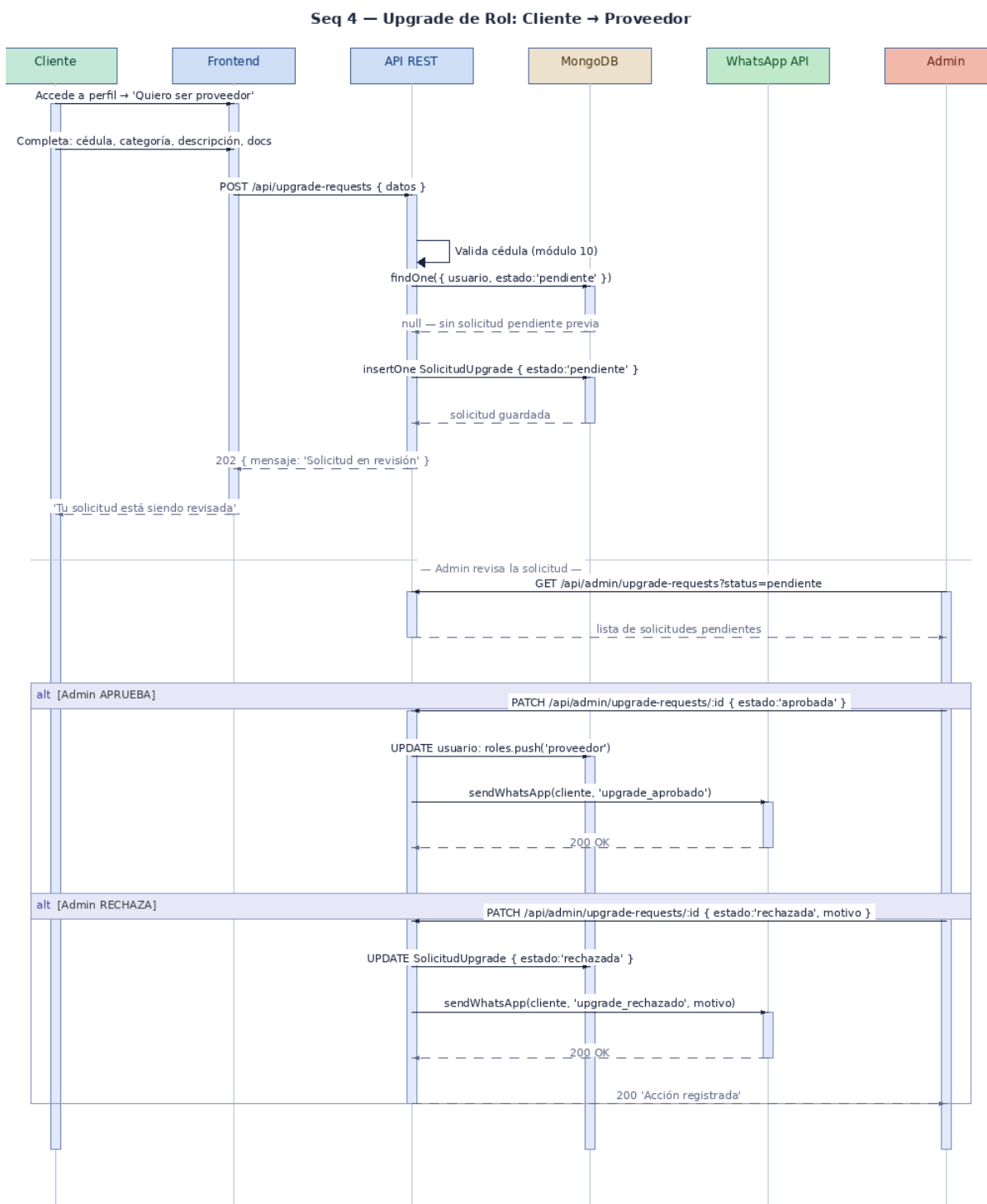


Figura 15

Diagrama de secuencia — Upgrade de rol: validación de cédula, notificación al admin y actualización de roles[] (RF-15, RF-16)



3.7.3. Diagrama de componentes

El diagrama de componentes (Figura 2, presentado en la sección 3.4) describe la arquitectura física del sistema. Sus interfaces principales son: navegador ↔ SPA de React (interacción directa), SPA ↔ API REST (HTTPS/REST + JWT Bearer), API REST ↔ MongoDB (TCP/Mongoose), API REST ↔ servicio WhatsApp ↔ Meta Cloud API (HTTPS/POST + Bearer), Multer ↔ almacenamiento de archivos (multipart/form-data), y Railway ↔ repositorio Git (pipeline continuo desde la rama principal). La disposición izquierda-derecha permite distinguir el flujo entre presentación, lógica de negocio, persistencia y servicios externos.

3.8. Restricciones técnicas del sistema

El SRS v3.1 establece restricciones técnicas que delimitan el espacio de diseño e implementación. La Tabla 4 resume las más relevantes.

Tabla 4

Restricciones técnicas del sistema ServiLink

Área	Restricción	Impacto en el diseño
Navegadores	Chrome 90+, Firefox 88+, Edge 90+, Safari 14+	React 18 + Vite para compatibilidad amplia con APIs web estables
Despliegue	Railway con pipeline continuo desde Git	Arquitectura unificada backend+frontend en un mismo servicio
Seguridad	HTTPS obligatorio con TLS válido	Certificado TLS gestionado automáticamente por Railway
Seguridad	bcrypt con factor de costo mínimo 10	Configuración fija en el middleware de registro

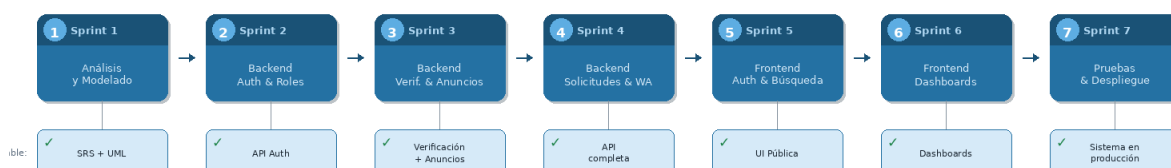
Seguridad	Variables sensibles en .env, nunca en el repositorio	dotenv en desarrollo; variables de entorno en Railway en producción
Validación	Cédula ecuatoriana: algoritmo dígito verificador módulo 10	Validación en el backend antes de crear el documento de usuario
Notificaciones	Número de teléfono debe estar activo en WhatsApp	Restricción informativa documentada en el formulario de registro
Interfaz	Responsiva desde 360 px de ancho	Diseño mobile-first con Tailwind CSS

3.9. Marco de desarrollo de software

A continuación, se describe el proceso de desarrollo organizado en siete sprints. Para cada sprint se describen las actividades de desarrollo realizadas y las pruebas ejecutadas al cierre de la iteración. La Figura 1 presentada en la sección 3.2 resume el plan general de sprints; a continuación, se detalla el contenido de cada uno.

Figura 16

Secuencia de los siete sprints del proyecto ServiLink con sus entregables verificables al cierre de cada iteración.



Sprint 1 — Análisis y modelado

Este sprint se orientó a definir con precisión qué debía hacer el sistema antes de escribir código. Se identificaron los tres actores principales y sus necesidades mediante análisis documental del contexto, definición de flujos y revisión con la directora del proyecto. Se elaboró el SRS con 21 RF y 19 RNF. Se diseñaron ocho diagramas de casos de uso, diez

diagramas de secuencia, un diagrama de clases y un diagrama de componentes. También se construyeron wireframes de las principales vistas del sistema: registro, buscador, perfil de proveedor, panel de cliente, panel de proveedor y panel de administrador.

Pruebas del sprint: revisión de consistencia entre los RF del SRS y los casos de uso UML, verificando que cada RF apareciera representado en al menos un diagrama. Los wireframes y la especificación fueron revisados con la directora del proyecto. Como resultado, el SRS y los diagramas UML quedaron establecidos como base para el desarrollo.

Sprint 2 — Backend: autenticación y gestión de usuarios

En este sprint se implementaron los módulos de registro, autenticación y gestión de perfiles que constituyen la base de acceso al sistema. Se inicializó el proyecto Node.js 20 con Express, dotenv y la estructura de directorios MVC (controllers/, routes/, models/, middlewares/). Se definió el esquema Mongoose del modelo User con el arreglo roles[], el campo verificationStatus, y la validación del dígito verificador módulo 10 de la cédula ecuatoriana implementada como función pura en utils/cedula.js. Se implementaron los endpoints POST /api/auth/register, POST /api/auth/register-provider, POST /api/auth/login (generación de JWT firmado con HS256), POST /api/auth/logout, GET /api/users/me y PATCH /api/users/me.

Pruebas del sprint: RF-01 registro exitoso de cliente (HTTP 201), RF-02 escenario A cédula inválida (HTTP 400), RF-02 escenario B cédula duplicada (HTTP 409), RF-02 escenario C registro exitoso de proveedor con verificationStatus 'pendiente' (HTTP 201), RF-03 inicio de sesión con JWT (HTTP 200), RF-04 cierre de sesión (HTTP 200), GET /users/me

sin campo password (HTTP 200), PATCH /users/me con updatedAt actualizado (HTTP 200). Todos los casos retornaron el código HTTP esperado.

Sprint 3 — Backend: verificación de proveedores y anuncios

En este sprint se implementaron los módulos que habilitan a un proveedor para publicar servicios y el flujo administrativo que controla ese acceso. Se implementó el middleware de Multer para carga de documentos en ruta no pública. Se desarrollaron los endpoints de verificación: POST /api/users/upload-docs (solo rol proveedor), GET /api/admin/verifications (solo rol admin) y PATCH /api/admin/verifications/:id (estados 'aprobar' o 'rechazar' con validación de valor). Se definió el esquema Announcement con los campos imagen[], estado ('activo'/'inactivo'/'suspendido') y referencia al proveedor. Se implementaron los endpoints de anuncios: GET /api/announcements (público, paginado), GET /api/announcements/:id, POST /api/announcements (solo proveedor verificado), PATCH y DELETE sobre /api/announcements/:id. Se implementó el endpoint POST /api/announcements/:id/report.

Pruebas del sprint: RF-05 cliente intenta subir documentos (HTTP 403), RF-06 admin visualiza verificaciones pendientes (HTTP 200), RF-07 escenario A acción inválida (HTTP 400), RF-07 escenario B aprobación exitosa con WhatsApp en consola (HTTP 200), RF-08 listado público paginado (HTTP 200), RF-19 escenario A cliente sin permiso (HTTP 403), RF-19 escenario B proveedor verificado crea anuncio con estado 'activo' (HTTP 201), RF-20 registro de reporte (HTTP 201). Todos los casos PASS.

Sprint 4 — Backend: solicitudes de servicio y notificaciones WhatsApp

En este sprint se implementó el flujo central de contratación y el servicio de notificaciones automáticas. Se encapsuló la integración con la WhatsApp Business Cloud API en `services/whatsapp/`, con la función de normalización de números de la forma `0XXXXXXXXXX` al formato E.164 (`+593XXXXXXXXXX`) en `lib/telefono.js`. Se implementó el middleware de autorización granular que verifica que el usuario autenticado sea el propietario de la solicitud (no solo que tenga rol proveedor). Se desarrollaron los endpoints: `POST /api/service-requests` (con restricción de no enviarse a sí mismo), `GET /api/service-requests/sent`, `PATCH /api/service-requests/:id/accept,/reject,/reschedule,/accept-reschedule` y `/reject-reschedule`. Se implementaron también los endpoints de upgrade de rol: `POST /api/upgrade-requests`, `GET /api/admin/upgrade-requests` y `PATCH /api/admin/upgrade-requests/:id`.

Pruebas del sprint: RF-10 E1 envío de solicitud (HTTP 201 con WhatsApp en consola), RF-11 E2a cliente intenta aceptar (HTTP 403), RF-11 E2b admin intenta aceptar (HTTP 403), RF-11 E2c proveedor dueño acepta (HTTP 200), RF-11 E4 reagendamiento (HTTP 200 con WhatsApp), RF-13 E5a aceptación del reagendamiento (HTTP 200). Todos los casos PASS. Se verificó el log de WhatsApp en consola para RF-07 y RF-12.

Sprint 5 — Frontend: autenticación y búsqueda

En este sprint se construyó la capa de presentación pública del sistema. Se inicializó el proyecto React 18 con Vite y se configuró Tailwind CSS con los breakpoints para responsividad desde 360 px. Se implementó el contexto de autenticación (`AuthContext`) para

gestionar el token JWT en el almacenamiento local del navegador y propagarlo a todos los componentes. Se desarrollaron las vistas de registro de cliente, registro de proveedor (con validación visual del campo cédula), inicio de sesión con redirección según roles[] y cierre de sesión.

Se construyó el módulo de búsqueda: campo de texto con filtro por categoría que consulta GET /api/announcements en tiempo real, resultados presentados como tarjetas con fotografía, nombre, categoría y badge de verificación del proveedor. Se desarrolló la vista de perfil público del proveedor con sus anuncios activos y el botón de solicitud de servicio. Se implementó el componente DateTimePicker para la selección de fecha y hora propuesta en el formulario de solicitud.

Pruebas del sprint: flujo completo de registro de cliente desde la interfaz, registro de proveedor con feedback visual de cédula inválida, inicio de sesión con redirección a dashboard de cliente y a dashboard de proveedor según roles[], búsqueda con filtro por categoría mostrando resultados correctos, envío de solicitud desde el perfil del proveedor con confirmación visual.

Sprint 6 — Frontend: dashboards de cliente, proveedor y administrador

En este sprint se implementaron los paneles de gestión para cada actor del sistema. Se construyó el layout compartido DashboardLayout con barra lateral responsiva que se colapsa en dispositivos móviles. El panel del cliente incluye la sección de solicitudes enviadas con estados visuales diferenciados por color (pendiente/aceptada/rechazada/reagendada) y los controles de respuesta al reagendamiento propuesto por el proveedor.

El panel del proveedor implementa tres secciones: gestión de anuncios con carga multi-imagen y control de estado activo/inactivo; solicitudes recibidas con los controles de aceptar, rechazar y reagendar; y perfil con los datos de verificación y el estado actual. El dashboard del usuario con rol dual muestra las secciones 'Como cliente' y 'Como proveedor' diferenciadas mediante pestañas, ocultando el botón de solicitud de servicio en el perfil propio.

El panel del administrador implementa cuatro módulos: estadísticas en tiempo real desde `/api/admin/stats`, lista de verificaciones pendientes con visualización de documentos y controles de aprobación/rechazo con comentario, anuncios reportados con las acciones de editar/suspender/desestimar, y gestión de usuarios con filtros por rol y estado.

Pruebas del sprint: navegación completa entre dashboards con token JWT activo, persistencia del token entre recargas de página, flujo completo de aceptación y rechazo de solicitudes desde el panel del proveedor, flujo de aprobación de verificación desde el panel del administrador, visualización correcta del dashboard dual sin botón de auto-solicitud.

Sprint 7 — Integración, pruebas y despliegue

En este sprint se validó el sistema completo y se dejó disponible en producción. Se ejecutó el plan de pruebas funcionales con Postman v11 sobre los módulos del sistema, completando 23 escenarios de prueba con resultado 23 PASS y 0 FAIL, que verifican explícitamente los 21 RF del SRS. Se documentó la API con Swagger UI (OAS 3.0) accesible en `/api-docs`, organizando los endpoints en seis grupos con esquemas de datos completos. Se construyó el directorio de producción del frontend con `npm run build` y se comprometió en el

repositorio para ser servido por Express en Railway. Se configuraron las variables de entorno en el panel de Railway (MONGODB_URI, JWT_SECRET, WhatsApp token y Phone Number ID) y se realizó el despliegue con pipeline continuo desde la rama main.

Pruebas del sprint: ejecución de los 23 escenarios de prueba funcionales con Postman v11 (resultados detallados en la Tabla 5 del Capítulo IV), verificación del health check en <https://servilink.up.railway.app/api/health>, verificación de la interfaz completa en producción desde tres navegadores distintos (Chrome, Firefox y Edge), y confirmación del pipeline continuo con un commit de corrección aplicado durante el despliegue.

Capítulo IV

Resultados de la investigación

4.1. Implementación del backend

El backend de ServiLink se implementó con Node.js 20 y Express, siguiendo el patrón de arquitectura MVC con separación de responsabilidades en directorios de controladores (controllers/), rutas (routes/), modelos (models/), middlewares (middlewares/) y servicios (services/whatsapp/). La API REST expone sus endpoints en seis grupos funcionales, documentados mediante Swagger UI (OAS 3.0) accesible en /api-docs. A continuación, se describe la implementación de cada módulo.

4.1.1. Módulo de autenticación y gestión de usuarios

El módulo de autenticación gestiona el ciclo completo de registro e inicio de sesión para los tres tipos de actor del sistema. El registro de clientes (RF-01) se implementa mediante el endpoint POST /api/auth/register, que valida el formato del correo electrónico, verifica su unicidad en la colección users de MongoDB, aplica hash bcrypt con factor de costo 10 a la contraseña y crea el documento de usuario con el arreglo roles: ['cliente']. Al completarse exitosamente, el sistema responde con HTTP 201 Created y devuelve un token JWT que incluye el identificador del usuario y sus roles asignados.

El registro de proveedores (RF-02) se implementa mediante POST /api/auth/register-provider e incorpora dos validaciones adicionales específicas del contexto ecuatoriano: la verificación del formato de diez dígitos del número de cédula y la aplicación del algoritmo de dígito verificador módulo 10 del Registro Civil. Si la cédula no supera alguna de estas

validaciones, el sistema responde con HTTP 400 Bad Request; si la cédula ya se encuentra registrada en otra cuenta, responde con HTTP 409 Conflict. Un registro exitoso crea el documento de usuario con roles: ['proveedor'] y verificationStatus: 'pendiente', indicando que la cuenta requiere aprobación administrativa antes de poder publicar anuncios.

El inicio de sesión (RF-03) se implementa en POST /api/auth/login, que verifica las credenciales contra la base de datos usando bcrypt.compare(), genera un JWT firmado con la clave secreta almacenada en variables de entorno y lo devuelve junto al perfil del usuario. El cierre de sesión (RF-04) se gestiona en el cliente mediante la eliminación del token; el endpoint POST /api/auth/logout confirma la operación con HTTP 200. La gestión del perfil propio se expone en GET /users/me y PATCH /users/me, asegurando que el campo password nunca aparezca en las respuestas de la API.

Figura 17

Swagger UI — Grupos Usuarios, Upgrade de rol y Solicitudes de servicio

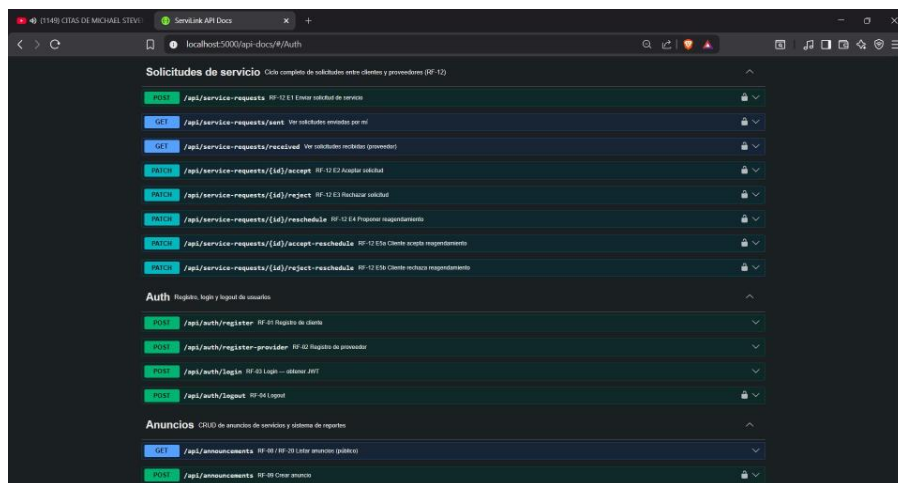


Figura 18

Swagger UI — Grupos Auth, Anuncios y Administración con endpoints expandidos

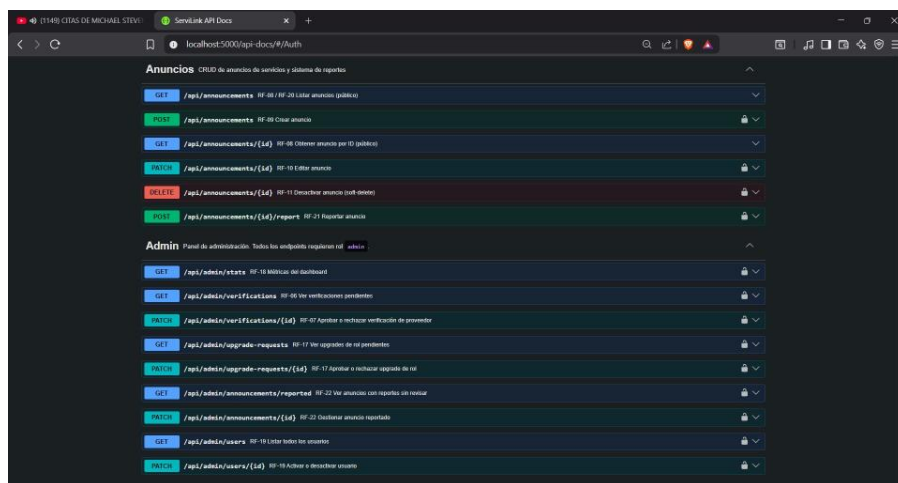
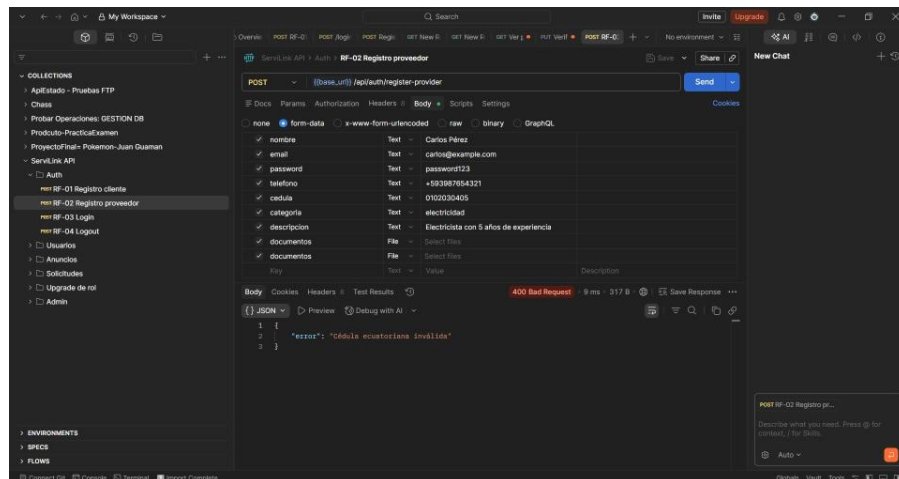


Figura 19

RF-01 — Registro exitoso de cliente María López. Respuesta HTTP 201 Created con token JWT y roles: ["cliente"]

RF-02 Escenario A — Cédula inválida (0102030405): el sistema retorna HTTP 400 Bad Request



RF-02 Escenario B — Cédula duplicada (0107190977 ya registrada): el sistema retorna HTTP 409 Conflict

nunca viajen por la red (HTTP 200). La actualización del perfil se implementa en PATCH /api/users/me: el endpoint permite modificar campos como nombre, teléfono y fotografía de perfil, y actualiza automáticamente el campo updatedAt del documento al momento de la escritura (HTTP 200).

Estos dos endpoints están protegidos por el middleware de autenticación JWT pero no tienen restricción de rol —son accesibles para clientes, proveedores y administradores por igual—, dado que la gestión del propio perfil es una funcionalidad universal del sistema independiente del rol del usuario.

Figura 23

GET /users/me — Perfil del usuario autenticado. El campo password no aparece en la respuesta (HTTP 200)

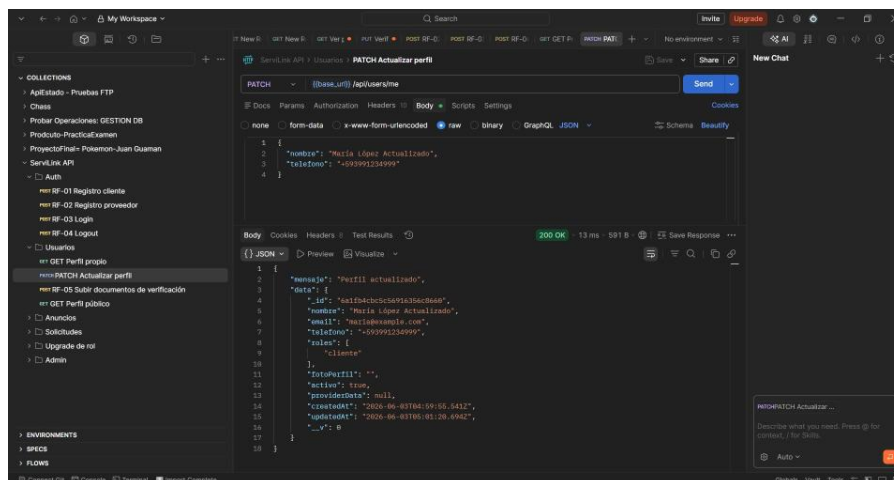
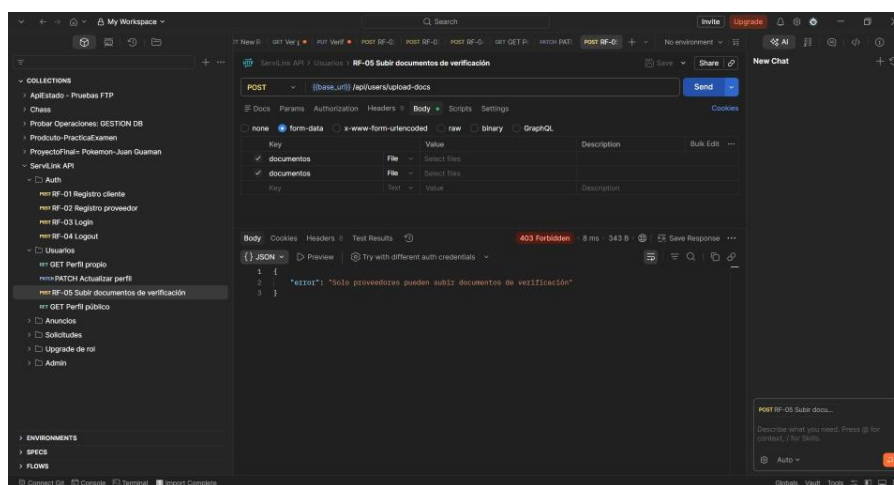


Figura 24

PATCH /users/me — Actualización de nombre y teléfono. El campo updatedAt se actualiza automáticamente (HTTP 200)



4.1.3. Módulo de verificación de proveedores

El módulo de verificación gestiona el flujo administrativo que habilita a un proveedor registrado para publicar anuncios en la plataforma. La carga de documentos de identidad (RF-

05) se implementa en POST `/api/users/upload-docs`, protegido por el middleware de autenticación JWT y un middleware de autorización por rol que restringe el acceso exclusivamente a usuarios con rol proveedor: un cliente que intente acceder a este endpoint recibe HTTP 403 Forbidden. Los archivos se procesan con Multer y se almacenan en una ruta no pública del servidor, accesible únicamente por el administrador autenticado (RNF-08).

La revisión administrativa de las verificaciones pendientes (RF-06) se expone en GET `/api/admin/verifications`, reservado al rol admin. El procesamiento de la decisión (RF-07) se implementa en PATCH `/api/admin/verifications/:id` y cubre dos escenarios: en el escenario A, si el campo estado recibe un valor distinto de 'aprobar' o 'rechazar', el sistema retorna HTTP 400 Bad Request con un mensaje que indica los valores permitidos, evitando actualizaciones con estados inválidos; en el escenario B, si el valor es válido, el sistema actualiza el campo `verificationStatus` del usuario —a 'aprobado' o a 'rechazado' según corresponda— y dispara el servicio de notificaciones WhatsApp para informar al proveedor del resultado con el comentario ingresado por el administrador.

Figura 25

RF-05 — Control de acceso: cliente intenta subir documentos y recibe HTTP 403 Forbidden

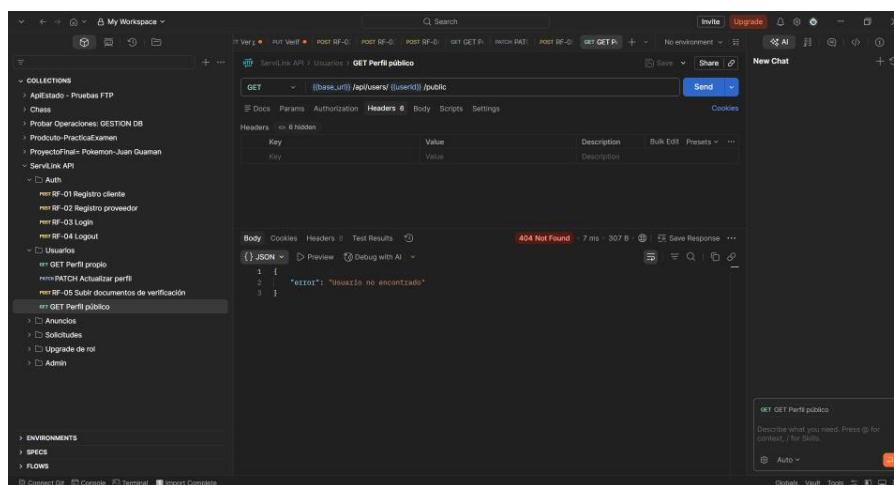


Figura 26

RF-06 — Admin consulta verificaciones pendientes. Carlos Pérez con verificationStatus: "pendiente" (HTTP 200)

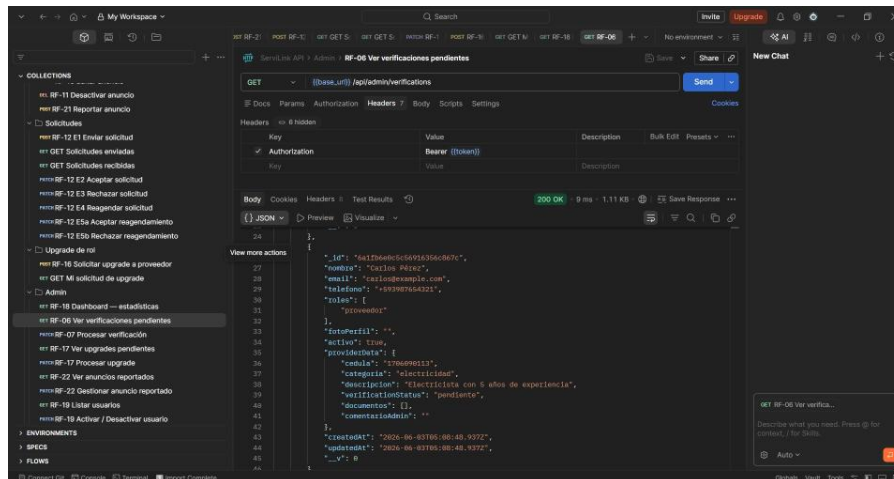


Figura 27

RF-07 Escenario A — Acción inválida: campo estado no permitido retorna HTTP 400 con lista de valores aceptados

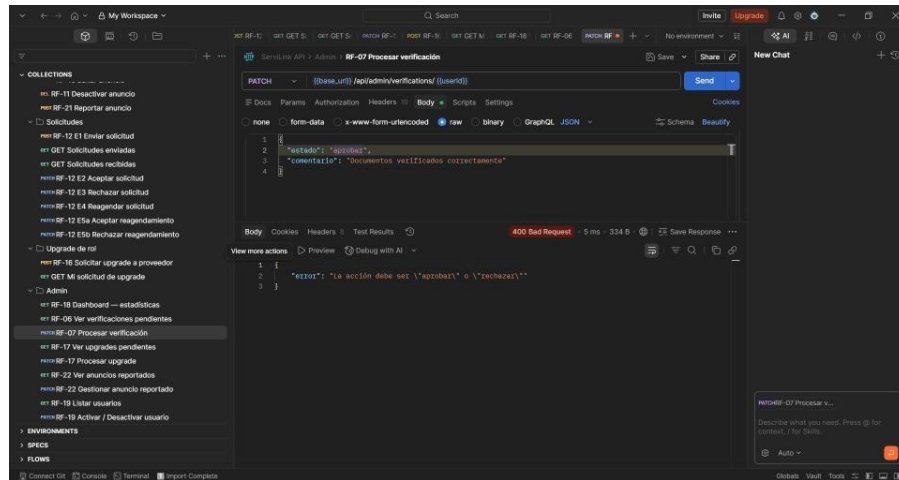
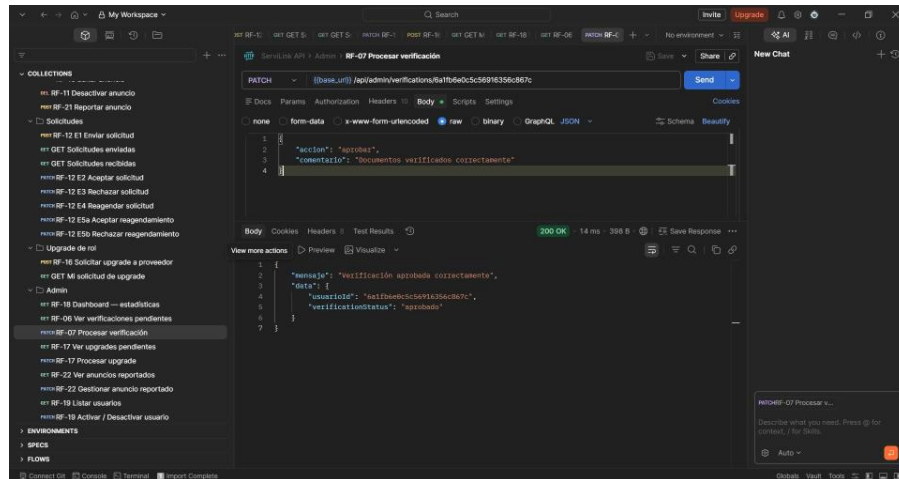


Figura 28

RF-07 Escenario B — Verificación aprobada: verificationStatus cambia a "aprobado" (HTTP 200). Log WhatsApp en consola



4.1.4. Módulo de anuncios

El módulo de anuncios gestiona el ciclo de vida completo de los servicios ofertados por los proveedores verificados. El listado público de anuncios (RF-08) se expone sin autenticación en GET /api/announcements, que retorna una lista paginada de anuncios con estado 'activo', incluyendo los datos del proveedor poblados mediante Mongoose. La consulta de un anuncio individual se expone en GET /api/announcements/:id. La creación de anuncios (RF-09) se

implementa en POST /api/announcements, protegido por un middleware que verifica simultáneamente que el token JWT sea válido, que el arreglo roles[] del usuario contenga 'proveedor' y que el campo verificationStatus sea 'aprobado': el incumplimiento de cualquiera de estas condiciones retorna HTTP 403 Forbidden.

Cada anuncio incluye título, descripción, categoría, precio referencial opcional y un arreglo de rutas de imágenes cargadas con Multer. Los anuncios pueden tener tres estados: 'activo' (visible en búsquedas), 'inactivo' (desactivado por el propio proveedor, no visible pero no eliminado) y 'suspendido' (desactivado por el administrador tras revisión de un reporte). La publicación de un anuncio exitoso retorna HTTP 201 Created con el documento completo y el estado inicial 'activo'.

Figura 29

RF-08 — Listado público paginado de anuncios activos sin autenticación (HTTP 200)

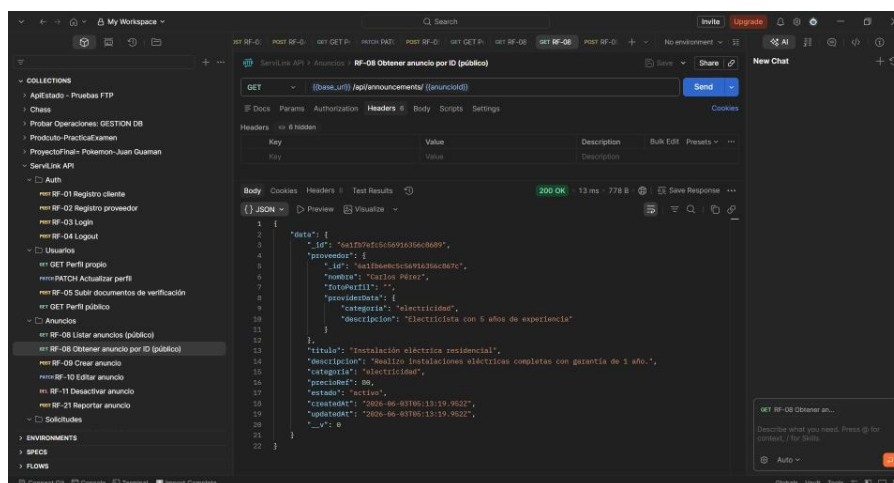


Figura 30

RF-19 Escenario A — Cliente intenta crear anuncio y recibe HTTP 403 Forbidden

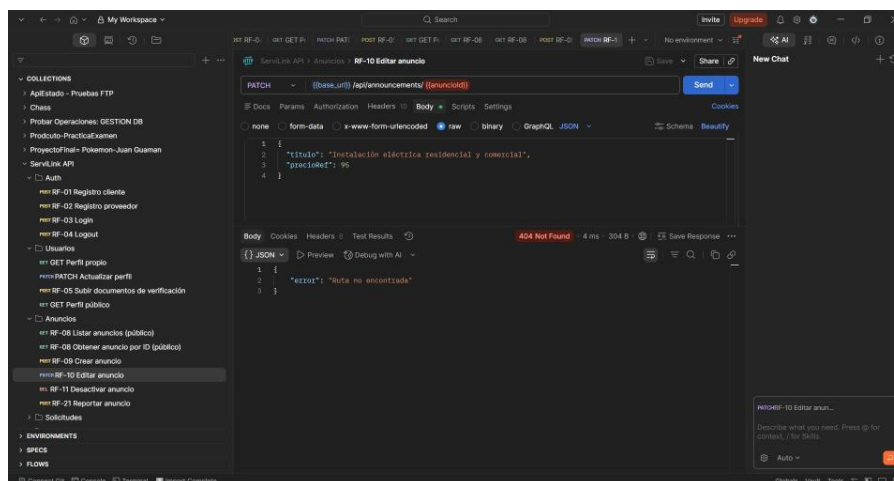
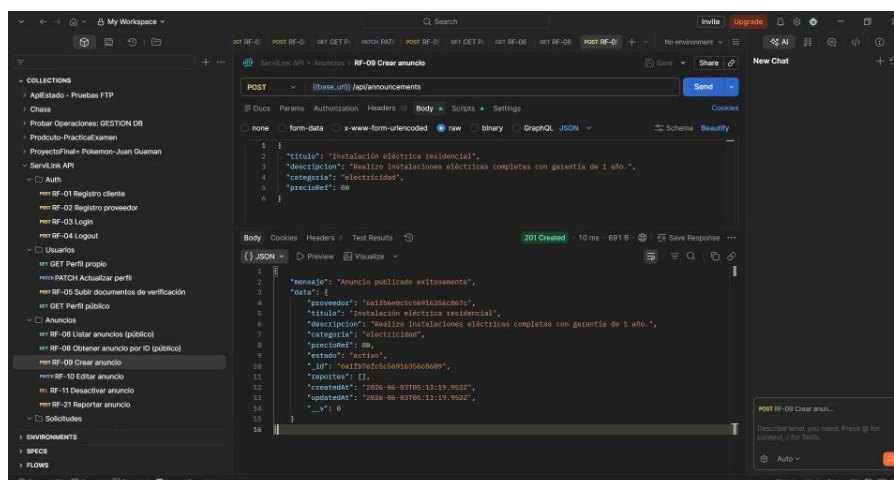


Figura 31

RF-19 Escenario B — Carlos (proveedor verificado) crea anuncio: HTTP 201 Created con estado: "activo"



4.1.5. Módulo de solicitudes de servicio

El módulo de solicitudes implementa el ciclo completo de contratación de un servicio: envío, aceptación, rechazo, reagendamiento y respuesta al reagendamiento. El envío de una solicitud (RF-10) se implementa en POST /api/service-requests: el sistema valida que el solicitante esté autenticado, que el proveedor destino tenga estado 'aprobado' y que el solicitante no sea el mismo usuario que el proveedor (regla de negocio del rol dual). La solicitud se crea con estado 'pendiente' y se dispara automáticamente una notificación WhatsApp al proveedor.

La gestión de solicitudes por el proveedor (RF-11) se expone en los endpoints PATCH /api/service-requests/:id/accept, /rejecty /reschedule. Un middleware de autorización granular verifica en cada uno de estos endpoints que el usuario autenticado sea específicamente el proveedor destinatario de la solicitud: un cliente que intente aceptar recibe HTTP 403, y un administrador que intente hacerlo también recibe HTTP 403, ya que solo el proveedor dueño puede gestionar sus solicitudes. La aceptación cambia el estado a 'aceptada' y establece el campo fechaConfirmada; el rechazo lo lleva a 'rechazada' con motivo opcional; el

reagendamiento lo pasa a 'reagendada' con la nueva propuesta de fecha, notificando al cliente por WhatsApp.

El seguimiento por parte del cliente (RF-13) se implementa mediante GET /api/service-requests/sent, que lista las solicitudes enviadas por el usuario autenticado con sus estados actuales. La respuesta al reagendamiento se implementa en PATCH /api/service-requests/:id/accept-reschedule y /reject-reschedule: si el cliente acepta, el estado vuelve a 'aceptada' con la nueva fechaConfirmada; si rechaza, el estado queda en 'rechazada'.

Figura 32

RF-10 Escenario 1 — María envía solicitud a Carlos: HTTP 201 Created, estado: "pendiente". Notificación WhatsApp al proveedor

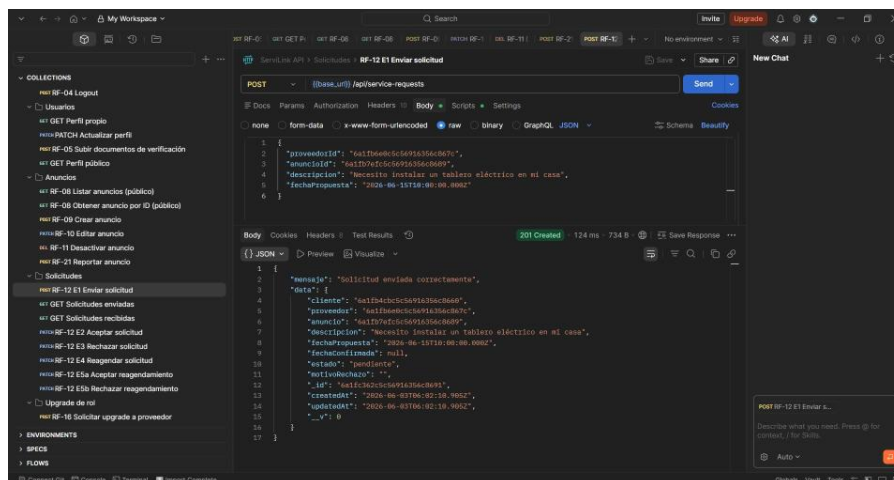


Figura 33

RF-11 Escenario 2a — Cliente intenta aceptar solicitud propia: HTTP 403 Forbidden

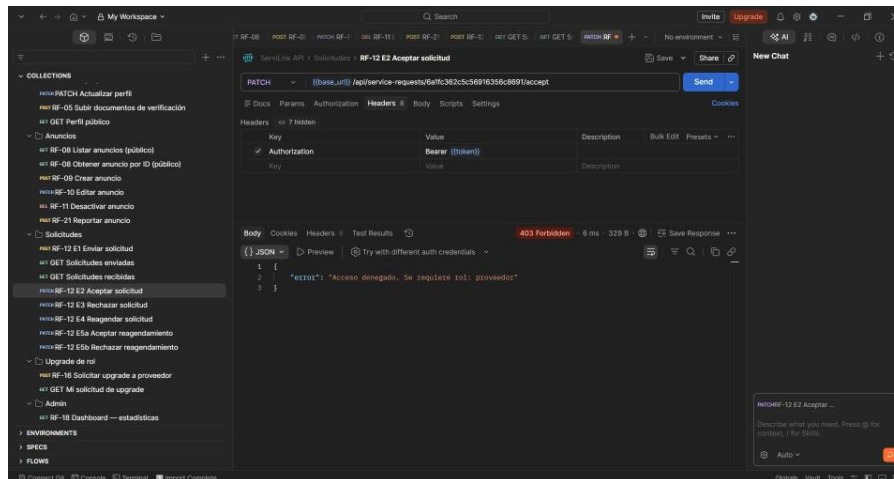


Figura 34

RF-11 Escenario 2b — Administrador intenta aceptar: HTTP 403 Forbidden

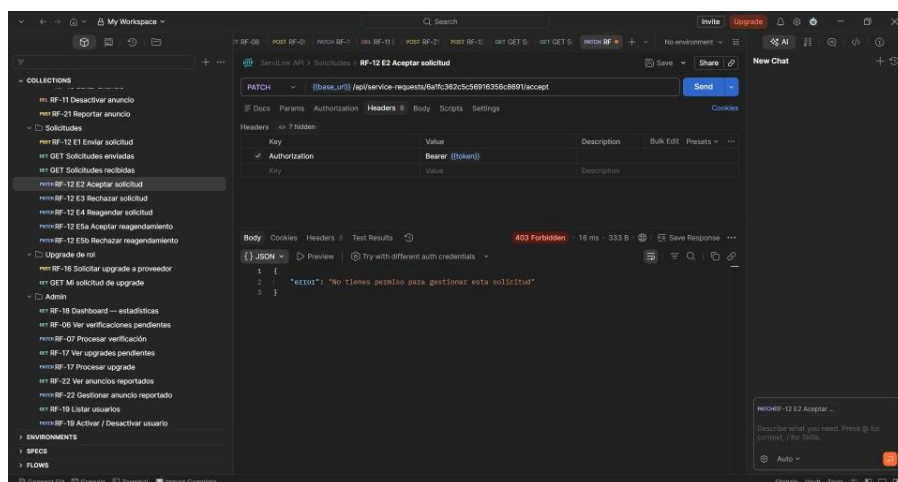


Figura 35

RF-11 Escenario 2c — Carlos (proveedor dueño) acepta: HTTP 200 OK, estado: "aceptada"

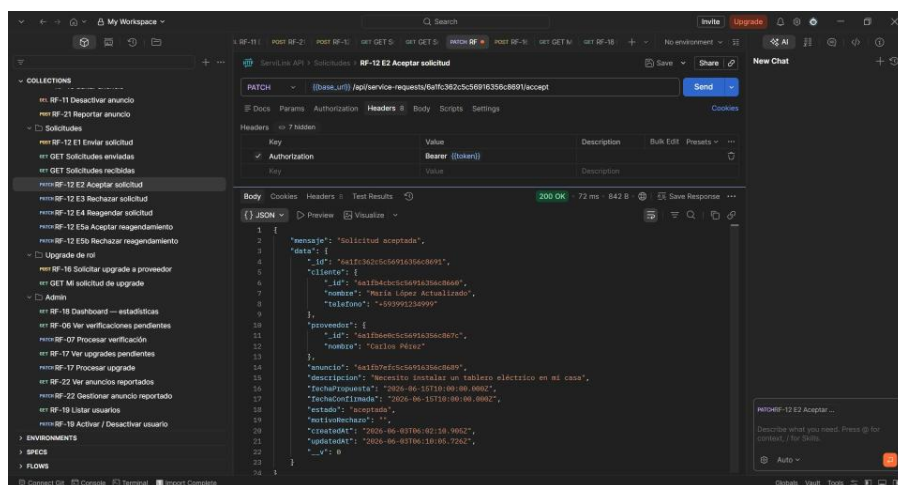


Figura 36

*RF-11 Escenario 4 — Carlos propone reagendamiento: HTTP 200, estado: "reagendada".
Notificación WhatsApp al cliente*

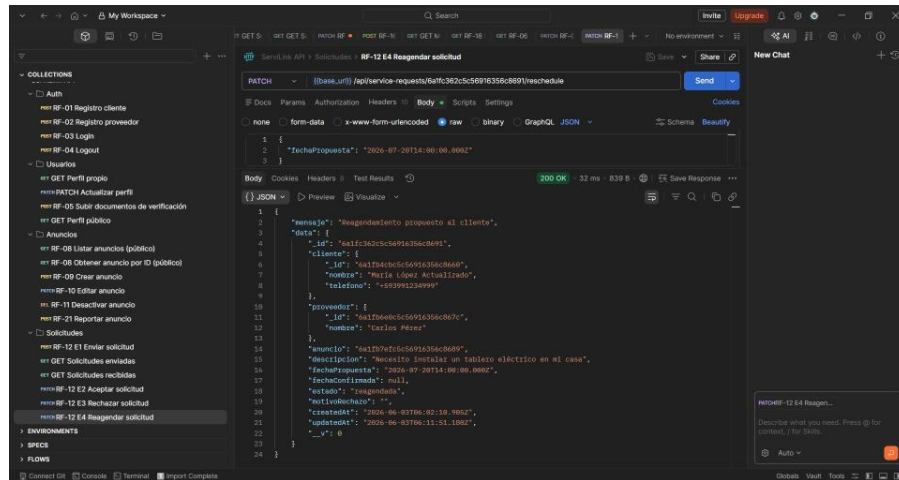
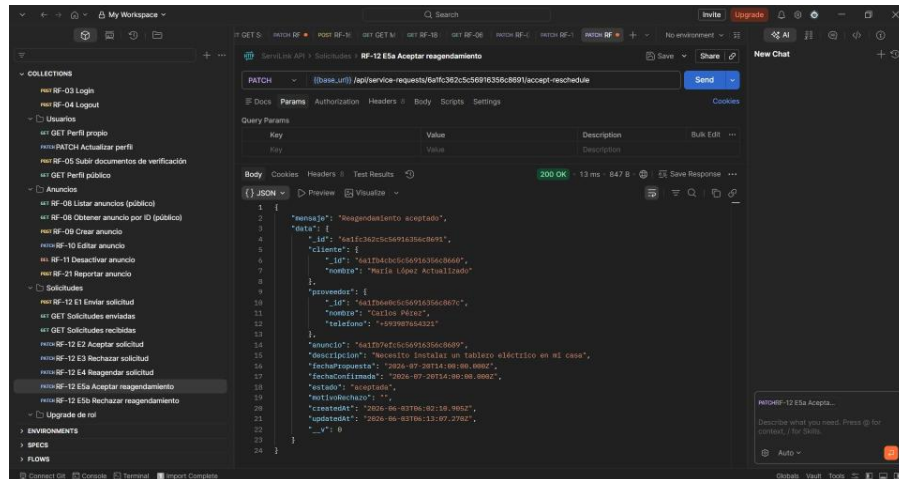


Figura 37

RF-13 Escenario 5a — María acepta el reagendamiento: HTTP 200, estado vuelve a "aceptada"



4.1.6. Módulo de rol dual y upgrade

El módulo de rol dual implementa el flujo que permite a un usuario con rol cliente solicitar la incorporación del rol proveedor a su cuenta existente, sin necesidad de crear una segunda cuenta. La solicitud de upgrade (RF-15) se implementa en POST /api/upgrade-requests: el sistema crea un documento en la colección upgrade_requests con estado 'pendiente' y notifica al administrador. La revisión por parte del administrador (RF-16) se expone en GET

/api/admin/upgrade-requests para listar las solicitudes pendientes, y en PATCH /api/admin/upgrade-requests/:id para resolverlas.

Al aprobar un upgrade, el sistema ejecuta una actualización atómica sobre el documento de usuario que añade 'proveedor' al arreglo roles[] si aún no está presente, y establece verificationStatus: 'pendiente' para que el usuario complete el proceso de verificación de identidad como proveedor. Al rechazar, el roles[] del usuario permanece sin cambios y se notifica al solicitante vía WhatsApp con el motivo del rechazo. Un proveedor que usa la plataforma como cliente (RF-14) puede hacerlo directamente desde su cuenta dual sin restricciones adicionales, con la única excepción de que no puede enviarse solicitudes de servicio a sí mismo.

4.1.7. Módulo de administración

El panel de administración (RF-17) se expone en GET /api/admin/stats, que retorna métricas en tiempo real del sistema: número total de usuarios por rol, proveedores con verificación pendiente, solicitudes de upgrade pendientes, solicitudes de servicio activas y anuncios con reportes pendientes de revisión. La gestión de usuarios (RF-18) se implementa en GET /api/admin/users, que admite filtros por rol y estado, y en PATCH /api/admin/users/:id/toggle-status para activar o desactivar cuentas. La moderación de anuncios reportados (RF-21) se expone en GET /api/admin/announcements/reported para listar los anuncios con reportes, y en endpoints específicos para editar, suspender o desestimar cada reporte.

Figura 38

RF-17 — Dashboard de estadísticas del administrador en tiempo real: 5 usuarios, 2 proveedores, 1 anuncio con reporte (HTTP 200)

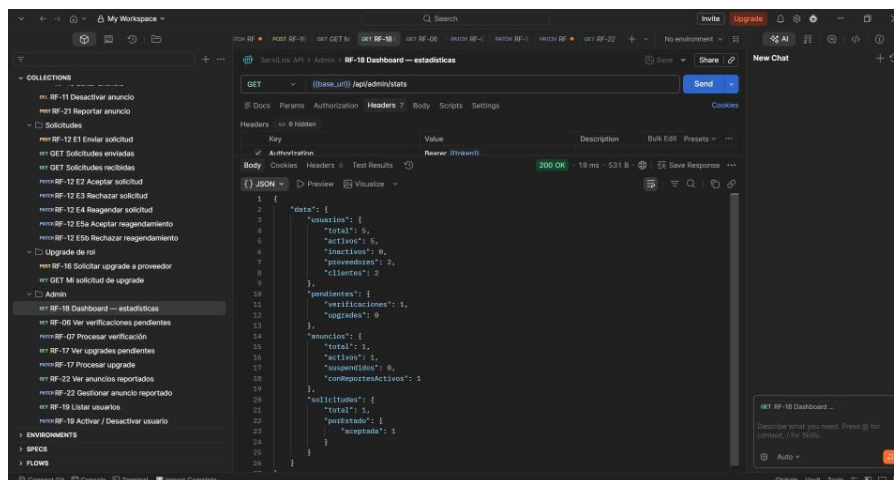


Figura 39

RF-21 — Admin visualiza anuncios reportados: anuncio de Carlos reportado por María (HTTP 200)

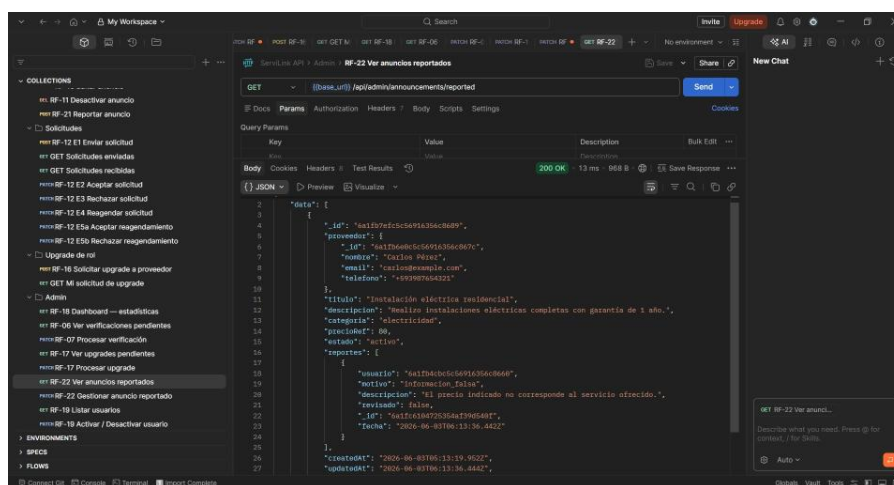
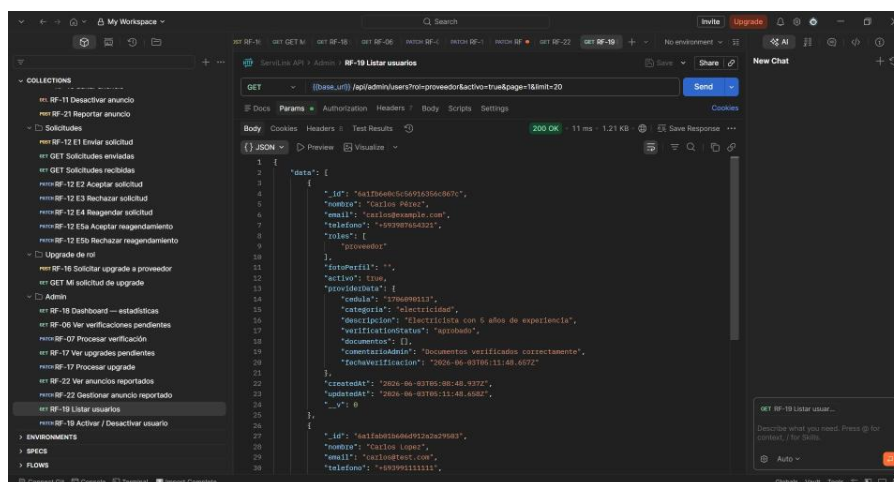


Figura 40

RF-18 — Admin lista proveedores activos filtrados por rol y estado (HTTP 200)



4.1.8. Servicio de notificaciones WhatsApp

El servicio de notificaciones se encapsula en el directorio `services/whatsapp/` y se integra con la WhatsApp Business Cloud API de Meta mediante peticiones HTTP POST al endpoint de la API de mensajería, autenticado con el token de acceso almacenado en variables de entorno. El servicio implementa la función de normalización de números telefónicos que convierte el formato local ecuatoriano (0XXXXXXXXXX) al formato internacional E.164 (+593XXXXXXXXXX) antes de cada envío. Se disparan notificaciones automáticas en cinco eventos del ciclo de vida del sistema: aprobación o rechazo de verificación de proveedor (RF-07), recepción de nueva solicitud de servicio (RF-12), confirmación o rechazo de solicitud, propuesta de reagendamiento y resultado del reagendamiento. En el entorno de desarrollo, el envío se registra en consola para trazabilidad sin consumir el cupo de la API.

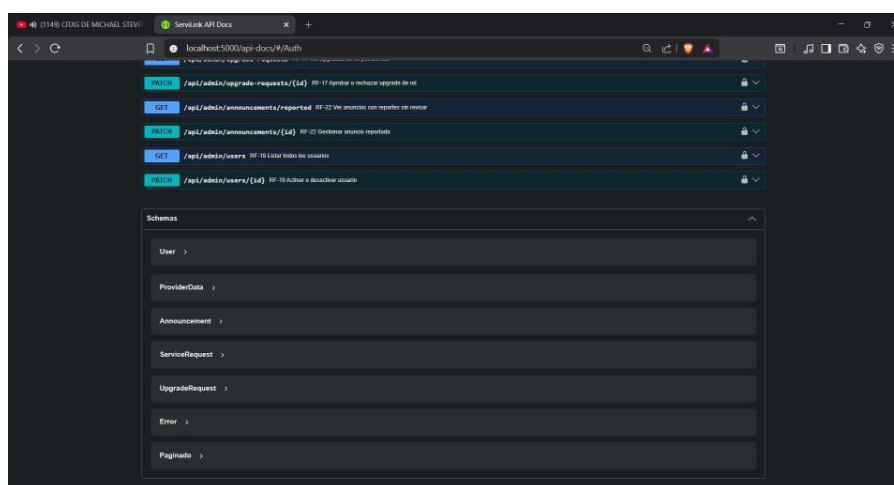
4.1.9. Documentación interactiva de la API

La API REST de ServiLink se documenta mediante Swagger UI (OAS 3.0), accesible en `http://localhost:5000/api-docs` durante el desarrollo. La documentación organiza los endpoints en seis grupos —Auth, Usuarios, Anuncios, Solicitudes de servicio, Upgrade de rol

y Administración— e incluye los esquemas de datos principales: User, ProviderData, Announcement, ServiceRequest y UpgradeRequest. Esta documentación interactiva permitió ejecutar y validar los casos de prueba directamente desde el navegador durante la fase de desarrollo, complementando las pruebas con Postman.

Figura 41

Swagger UI — Vista completa del grupo Administración con todos los endpoints y esquemas de datos



4.2. Implementación del frontend

El frontend de ServiLink se implementó como una Single Page Application (SPA) con React 18, Vite como herramienta de construcción y Tailwind CSS como framework de estilos. La aplicación gestiona el estado de autenticación y los datos de sesión a través del contexto de React, almacenando el token JWT en el almacenamiento local del navegador. La interfaz está completamente en español y es responsiva desde 360 px de ancho (RNF-10 y RNF-13). El directorio de producción generado por Vite (/dist) se sirve desde el mismo servicio Express del backend en Railway, usando la ruta raíz para la SPA con fallback a index.html.

4.2.1. Módulo de autenticación y registro

Las vistas de registro e inicio de sesión implementan formularios con validación en tiempo real que muestran mensajes de error descriptivos junto al campo correspondiente (RNF-11). El flujo de registro no supera los tres pasos requeridos por RNF-12: selección de tipo de cuenta, formulario de datos y confirmación. El formulario de registro de proveedor incluye el campo de cédula con validación visual que informa al usuario sobre el formato esperado antes del envío al servidor. Tras el inicio de sesión exitoso, la aplicación decodifica el JWT para determinar los roles del usuario y redirige al dashboard correspondiente.

4.2.2. Panel del cliente

El panel del cliente agrupa las funcionalidades de búsqueda y contratación de servicios. El módulo de búsqueda (RF-08) implementa un campo de texto con filtro por categoría que consulta en tiempo real el endpoint GET /api/announcements: los resultados se presentan como tarjetas con la foto, nombre, categoría y badge de verificación del proveedor. Al seleccionar un proveedor, la vista de perfil público (RF-09) muestra sus anuncios activos y el botón de solicitud de servicio. El formulario de solicitud (RF-10) incluye un campo de descripción y un selector de fecha y hora propuesta implementado con el componente DateTimePicker. La sección de seguimiento (RF-13) lista las solicitudes enviadas con su estado actual (pendiente, aceptada, rechazada, reagendada) y la opción de responder al reagendamiento propuesto por el proveedor.

4.2.3. Panel del proveedor

El panel del proveedor organiza sus funcionalidades en tres secciones diferenciadas: gestión de anuncios, solicitudes recibidas y perfil. La galería de anuncios (RF-19) permite crear

nuevos anuncios con carga de múltiples imágenes, editar los existentes y cambiar su estado entre activo e inactivo. La sección de solicitudes recibidas (RF-11) lista las solicitudes pendientes con los datos del cliente solicitante y la fecha/hora propuesta, y ofrece los controles de aceptar, rechazar y reagendar. El dashboard del usuario con rol dual muestra claramente las secciones 'Como cliente' y 'Como proveedor' diferenciadas visualmente mediante pestañas, cumpliendo con RNF-14.

4.2.4. Panel del administrador

El panel de administración centraliza las funciones de moderación y gestión del sistema. La sección de verificaciones muestra los proveedores con solicitud pendiente, permite revisar los documentos cargados y ejecutar la decisión de aprobación o rechazo con comentario opcional. La sección de anuncios reportados presenta cada anuncio con la lista de reportes recibidos y sus motivos, y ofrece los controles de editar contenido, suspender anuncio o desestimar el reporte. La sección de gestión de usuarios permite listar, filtrar por rol y estado, y activar o desactivar cuentas. El dashboard inicial presenta las métricas estadísticas del sistema obtenidas del endpoint `/api/admin/stats`.

4.3. Despliegue en producción

El despliegue de ServiLink se realizó en Railway, con un único servicio que aloja tanto el backend Node.js como el frontend compilado. El pipeline de despliegue continuo se configuró conectando directamente el repositorio Git (rama main) al servicio de Railway: cada push a la rama principal desencadena automáticamente un nuevo despliegue, garantizando que la versión en producción siempre corresponda al estado del repositorio (RNF-19).

La estrategia de despliegue unificado —backend y frontend en el mismo servicio— se implementó compilando el frontend con Vite (`npm run build`, que genera el directorio `/dist`) y comprometiendo ese directorio en el repositorio. El servidor Express sirve los archivos estáticos desde `/dist` mediante el middleware `express.static()`, y redirige cualquier ruta no reconocida a `index.html` para que la SPA de React gestione el enrutamiento en el cliente. La variable de entorno `VITE_API_URL` se establece como cadena vacía para que todas las peticiones de la API usen rutas relativas al mismo dominio, evitando problemas de CORS en producción. La ruta `GET /api/health` actúa como health check del servicio en Railway y retorna el estado de la conexión a MongoDB.

Las variables sensibles del sistema —cadena de conexión MongoDB con el parámetro `/servilink?authSource=admin`, `JWT_SECRET`, token de acceso de WhatsApp Business API y Phone Number ID— se gestionan íntegramente como variables de entorno en el panel de Railway y nunca se exponen en el repositorio (RNF-16). La plataforma desplegada es accesible en línea en `https://servilink.up.railway.app`, con certificado TLS proporcionado automáticamente por Railway (RNF-04).

4.4. Pruebas funcionales

4.4.1. Entorno y herramientas de prueba

Las pruebas se ejecutaron en junio de 2026 mediante Postman v11 y Swagger UI sobre la API REST en Node.js 20 y Express, conectada a MongoDB 7 en Docker. El informe independiente de pruebas registra 23 escenarios ejecutados, todos con resultado PASS, incluyendo respuestas exitosas y controles negativos de autorización, validación de cédula,

duplicidad, propiedad de recursos y roles. El documento aporta evidencia de integración entre API y base de datos, autenticación JWT, control RBAC, persistencia, carga de documentos y emisión de notificaciones de WhatsApp en modo de desarrollo (ServiLink, 2026).

La cobertura debe interpretarse por trazabilidad y no únicamente por cantidad de filas: varios requerimientos se verifican dentro de un mismo flujo o escenario. Por ejemplo, el ciclo de solicitudes comprueba creación, aceptación, reagendamiento, respuesta y notificaciones; y el módulo administrativo comprueba estadísticas, usuarios y anuncios reportados. No se realizaron pruebas automatizadas de carga, penetración o disponibilidad prolongada, por lo que los RNF asociados se reportan como diseñados o pendientes, según corresponda.

Además de comprobar las respuestas de cada endpoint, la ejecución manual permitió verificar la integración entre componentes en los flujos completos: persistencia y consulta de datos en MongoDB después de operaciones de la API; aplicación del token JWT y de los controles de autorización entre módulos; carga y recuperación de archivos; y emisión de notificaciones de WhatsApp en las transiciones correspondientes. Estas comprobaciones son evidencias de integración funcional de extremo a extremo, aunque no sustituyen una suite automatizada de pruebas de integración, cuya ausencia se reconoce como limitación del proyecto.

Las pruebas funcionales se ejecutaron sobre el entorno de desarrollo local con el servidor backend en Node.js 20 + Express corriendo en el puerto 5000 y la base de datos MongoDB 7 ejecutándose en un contenedor Docker en la dirección `mongodb://localhost:27017/servilink`. Las herramientas utilizadas fueron Postman v11 con la colección ServiLink API v1.0 para la ejecución de los casos de prueba, y Swagger UI (OAS

3.0) en `http://localhost:5000/api-docs` para la verificación previa de la documentación interactiva. Se utilizaron tres usuarios de prueba representativos de los actores del sistema:

- `admin@servilink.com` — rol: administrador (creado con script `seed-admin.js`)
- `maria@example.com` — rol: cliente (registro directo mediante la API)
- `carlos@example.com` — rol: proveedor, cédula: 1706090113, estado: verificado

4.4.2. Resultados por módulo

Las pruebas se organizaron siguiendo la misma estructura de módulos del SRS v3.1. En el módulo de autenticación se verificaron los cuatro requerimientos (RF-01 a RF-04) mediante cinco escenarios: registro exitoso de cliente (HTTP 201), rechazo de cédula inválida (HTTP 400), rechazo de cédula duplicada (HTTP 409), registro exitoso de proveedor (HTTP 201 con `verificationStatus` 'pendiente'), inicio de sesión con generación de JWT (HTTP 200) y cierre de sesión (HTTP 200). Todos los escenarios retornaron el código HTTP esperado y el comportamiento documentado en el SRS.

En el módulo de verificación de proveedores se verificaron RF-05, RF-06 y RF-07, comprobando el control de acceso por roles (HTTP 403 para clientes que intentan subir documentos), la correcta visualización de la lista de verificaciones pendientes y el procesamiento de la decisión administrativa con los dos flujos posibles (aprobar y rechazar). La notificación WhatsApp asociada a RF-07 se verificó mediante el registro en consola del entorno de desarrollo, confirmando que el mensaje se construye y envía correctamente.

Los módulos de búsqueda y anuncios cubrieron RF-08, RF-19, RF-20 y RF-21, verificando el acceso público al listado paginado, el control de acceso para la creación (solo proveedores verificados), la publicación exitosa con estado 'activo', el registro de un reporte por contenido inapropiado y la consulta administrativa de anuncios reportados. El módulo de solicitudes de servicio fue el más extenso, con siete escenarios (RF-10, RF-11 y RF-13) que cubrieron el ciclo completo: envío de solicitud, intentos de aceptación por roles no autorizados (HTTP 403 para cliente y administrador), aceptación por el proveedor dueño (HTTP 200), reagendamiento con propuesta de nueva fecha y aceptación del reagendamiento por el cliente. El módulo de administración verificó RF-17 (estadísticas en tiempo real), RF-18 (filtrado de usuarios por rol y estado) y RF-21 (visualización de anuncios reportados).

4.4.3. Tabla de resultados

La Tabla 5 presenta el resumen completo de los 23 escenarios de prueba ejecutados, con los endpoints, métodos HTTP, códigos de respuesta esperados y resultados obtenidos.

Tabla 5

Resumen de resultados de pruebas funcionales de ServiLink

RF	Escenario	Método	Endpoint	HTTP esperado	Resultado
RF-01	Registro cliente	POST	/api/auth/register	201 Created	PASS
RF-02	Cédula inválida	POST	/api/auth/register-provider	400 Bad Request	PASS
RF-02	Cédula duplicada	POST	/api/auth/register-provider	409 Conflict	PASS

RF-02	Registro proveedor	POST	/api/auth/register-provider	201 Created	PASS
RF-03	Inicio de sesión	POST	/api/auth/login	200 OK	PASS
RF-04	Cierre de sesión	POST	/api/auth/logout	200 OK	PASS
RF-05	Docs — cliente (403)	POST	/api/users/upload-docs	403 Forbidden	PASS
RF-06	Ver verificaciones	GET	/api/admin/verifications	200 OK	PASS
RF-07 E-A	Acción inválida	PATCH	/api/admin/verifications/:id	400 Bad Request	PASS
RF-07 E-B	Aprobar verificación	PATCH	/api/admin/verifications/:id	200 OK	PASS
RF-08	Listar anuncios	GET	/api/announcements	200 OK	PASS
RF-08	Anuncio por ID	GET	/api/announcements/:id	200 OK	PASS
RF-19	Crear — sin rol (403)	POST	/api/announcements	403 Forbidden	PASS
RF-19	Crear — proveedor	POST	/api/announcements	201 Created	PASS
RF-10	Enviar solicitud	POST	/api/service-requests	201 Created	PASS
RF-11	Aceptar — cliente (403)	PATCH	/api/service-requests/:id/accept	403 Forbidden	PASS
RF-11	Aceptar — admin (403)	PATCH	/api/service-requests/:id/accept	403 Forbidden	PASS
RF-11	Aceptar — proveedor	PATCH	/api/service-requests/:id/accept	200 OK	PASS

RF-11	Reagendar	PATCH	/api/service-requests/:id/reschedule	200 OK	PASS
RF-13	Aceptar reagendamiento	PATCH	/api/service-requests/:id/accept-reschedule	200 OK	PASS
RF-17	Dashboard stats	GET	/api/admin/stats	200 OK	PASS
RF-18	Listar usuarios	GET	/api/admin/users	200 OK	PASS
RF-20	Reportar anuncio	POST	/api/announcements/:id/report	201 Created	PASS
RF-21	Anuncios reportados	GET	/api/admin/announcements/report	200 OK	PASS

4.5. Análisis de resultados

La ejecución de 23 escenarios de prueba funcionales con Postman arrojó un resultado de 23 PASS y 0 FAIL; es decir, el 100 % de los casos ejecutados resultó satisfactorio. Estos casos verifican de forma explícita los 21 requerimientos funcionales mediante trazabilidad directa o agrupada, concentrándose en los módulos de autenticación, verificación, búsqueda, solicitudes, anuncios y administración. Los requerimientos RF-09 (perfil público), RF-12 (notificaciones de eventos de cita), RF-14 (proveedor actuando como cliente), RF-15 (solicitud de upgrade de rol) y RF-16 (revisión administrativa del upgrade) se validaron durante el uso funcional del sistema o de manera indirecta mediante los flujos asociados, aunque no cuentan con un caso de prueba independiente en la Tabla 5; su verificación formal individual se plantea como trabajo futuro. Este resultado verifica el cumplimiento del cuarto objetivo específico del proyecto y permite establecer la correspondencia entre los objetivos planteados en el Capítulo I y los resultados obtenidos.

El primer objetivo específico —realizar el análisis de requerimientos— se materializó en el SRS v3.1, que identificó los 21 RF y 19 RNF del sistema. La trazabilidad entre requerimientos y pruebas, evidenciada en la Tabla 5, confirma que el análisis fue suficientemente completo para guiar el desarrollo y la validación del sistema.

El segundo objetivo específico —diseñar la arquitectura, el modelo de datos y la interfaz— se concretó en ocho diagramas de casos de uso, diez diagramas de secuencia, el diagrama de clases y el diagrama de componentes. La arquitectura cliente-servidor en tres capas diseñada en el Capítulo III se implementó sin desviaciones significativas respecto al diseño planteado.

El tercer objetivo específico —implementar los módulos funcionales— se evidencia en el conjunto de endpoints que cubren la autenticación con roles, la verificación de proveedores, la búsqueda, la gestión de solicitudes con propuesta de citas y el sistema de notificaciones WhatsApp. Un aspecto relevante del análisis de resultados es la robustez del control de acceso basado en roles: los escenarios de gestión de solicitudes (RF-11) que prueban intentos de aceptación por roles no autorizados (cliente y administrador) retornaron correctamente HTTP 403, demostrando que el middleware de autorización granular funciona según lo especificado en el diseño.

El cuarto objetivo específico —validar y desplegar el sistema— queda demostrado por la ejecución satisfactoria de las pruebas funcionales y la disponibilidad de la plataforma en <https://servilink.up.railway.app>. La elección de Railway como plataforma de despliegue con

pipeline continuo desde Git garantiza que las correcciones aplicadas tras las pruebas se reflejan automáticamente en producción sin intervención manual, alineándose con RNF-19.

Como observación general del proceso de implementación, el módulo que presentó mayor complejidad fue el de solicitudes de servicio (RF-10 a RF-13), dada la necesidad de coordinar estados entre múltiples colecciones, validar autorizaciones granulares por propietario de recurso (no solo por rol genérico) e integrar notificaciones WhatsApp en cada transición de estado. Esta complejidad se refleja en que la gestión de solicitudes (RF-10, RF-11 y RF-13) concentra siete de los 23 escenarios de prueba ejecutados, el mayor número de escenarios de cualquier grupo funcional del sistema.

Cabe señalar como limitación de esta fase que la validación descrita se centró en los requerimientos funcionales (RF) mediante pruebas de caja negra sobre la API. Los requerimientos no funcionales definidos en la Tabla 2, incluido el tiempo de respuesta de la API (RNF-01, ≤ 2 s), no fueron perfilados de manera sistemática con una herramienta de medición de rendimiento durante esta fase del proyecto, por lo que su cumplimiento no puede darse por verificado con evidencia empírica y se identifica como una limitación del alcance de las pruebas realizadas.

4.6. Validación con usuarios finales

La validación con usuarios finales se realizó de manera exploratoria mediante demostraciones presenciales y conversaciones informales —principalmente por recomendación boca a boca— con 30 personas que representaban perfiles potenciales de clientes y proveedores. Cada participante recorrió las funciones principales con

acompañamiento del autor: registro, búsqueda, publicación, solicitud, aceptación o reagendamiento y consulta del panel correspondiente.

La retroalimentación no se obtuvo mediante una encuesta estructurada ni una escala estandarizada. Se registraron verbalmente las dificultades, dudas y sugerencias que se repetían durante las demostraciones; posteriormente se agruparon por módulo y se tradujeron en acciones de mejora. Para fortalecer la trazabilidad, la Tabla 6 conserva la observación reportada y la modificación aplicada. Este procedimiento permite documentar aceptación cualitativa y detectar problemas evidentes de interacción, pero no permite calcular indicadores estadísticos de usabilidad ni generalizar resultados a la población de Cuenca.

La validación se considera complementaria a las pruebas funcionales: las pruebas de Postman comprueban el comportamiento de la API, mientras que las demostraciones y conversaciones aportan evidencia cualitativa sobre comprensión de pantallas y recorrido de tareas. Como trabajo futuro se propone aplicar un instrumento estandarizado —por ejemplo, la escala SUS— y registrar tasa de finalización, tiempo y errores por tarea en un piloto con selección definida de participantes.

Las observaciones recogidas se clasificaron por módulo y se priorizaron según su impacto en la experiencia de uso. La Tabla 6 presenta las observaciones reportadas y la acción aplicada en cada caso.

Tabla 6

Observaciones de los usuarios durante la prueba de usabilidad y acciones aplicadas

N.º	Módulo o pantalla	Observación reportada	Acción aplicada
1	Perfil de usuario	Los usuarios no identificaban cómo cambiar su fotografía de perfil; el control de carga no resultaba evidente y la imagen se mostraba en un tamaño reducido.	Se rediseñó el componente de perfil: la fotografía se muestra en tamaño ampliado, con una capa superpuesta e ícono de cámara que abre el selector de archivo al hacer clic, una indicación de los formatos admitidos y el tamaño máximo, y una vista previa inmediata antes de confirmar la carga.
2	Panel de administración	Los colores empleados en las tarjetas de estadísticas dificultaban distinguir la información relevante y resultaban poco legibles.	Se ajustó la paleta de colores del panel, aumentando el contraste entre el fondo y el texto de los indicadores.
3	Panel de administración	Al consultar el listado de usuarios no era posible ver la fotografía de cada cuenta ni acceder al detalle de un proveedor y sus anuncios.	Se incorporó la fotografía en el listado de usuarios y una vista de detalle del proveedor que muestra sus anuncios publicados y los pendientes de aprobación.
4	Búsqueda por categoría	El selector de categorías era una lista fija y poco flexible; además, los íconos no se visualizaban correctamente al cargar la página.	Se sustituyó el selector por un campo de búsqueda con autocompletado e íconos representativos de cada profesión, y se migraron los íconos a recursos alojados localmente para garantizar su carga.
5	Galería de anuncios	El desplegable de categorías se recortaba al abrirse sobre la sección superior de la página y el desplazamiento interno se perdía al navegar por las opciones.	Se reestructuró la sección superior para permitir que el desplegable se despliegue fuera de sus límites y se corrigió el comportamiento del

			desplazamiento dentro de la lista.
6	Solicitud de servicio	El selector de fecha y hora se recortaba al abrirse cerca del borde inferior de la pantalla, impidiendo seleccionar la cita.	Se reposicionó el componente para que se despliegue hacia arriba cuando no dispone de espacio inferior suficiente.
7	Registro de proveedor	El campo de teléfono generaba confusión respecto al formato esperado, en particular sobre el uso del código de país.	Se incorporó el prefijo +593 de forma automática y se habilitó el ingreso del número en formato local, normalizándolo antes de almacenarlo.
8	Perfil público del proveedor	Al acceder a la vista de perfil completo no se mostraba información suficiente sobre el proveedor.	Se completó la vista con los datos del proveedor, los servicios realizados y el espacio destinado a las valoraciones recibidas.
9	Solicitud de ascenso a proveedor	El formulario solicitaba nuevamente datos que el usuario ya había registrado al crear su cuenta.	Se habilitó el autocompletado del formulario con la información existente del usuario, reduciendo el número de campos a llenar.
10	Gestión de solicitudes	Una vez aceptada la solicitud, los usuarios no disponían de una vía directa para coordinar los detalles del servicio.	Se incorporó un acceso directo de contacto por WhatsApp en las solicitudes aceptadas.

Las observaciones se concentraron en aspectos de interfaz y de experiencia de uso, y no en fallos funcionales, lo que resulta coherente con el resultado de las pruebas presentadas en la Tabla 5. La totalidad de las observaciones recogidas fue incorporada al sistema mediante ajustes en los componentes correspondientes, y cada modificación se verificó nuevamente sobre el entorno de producción antes de darla por concluida. La incorporación de estas mejoras permitió ajustar la plataforma a las expectativas de sus usuarios finales antes de la entrega definitiva.

4.7. Conclusiones

El desarrollo de ServiLink permitió cumplir el objetivo general del proyecto — desarrollar una plataforma web para la oferta y contratación de servicios independientes con verificación de proveedores en la ciudad de Cuenca— por medio del análisis, el diseño y la construcción de sus módulos funcionales, así como su despliegue en un entorno de producción en la nube. Con base en los resultados presentados en el Capítulo IV, se establecen las siguientes conclusiones, organizadas en correspondencia con los objetivos específicos y las preguntas de investigación planteados en el Capítulo I.

En relación con el primer objetivo específico, el análisis de requerimientos se concretó en el documento de especificación de requerimientos de software SRS v3.1, que identificó 21 requerimientos funcionales y 19 requerimientos no funcionales a partir de las necesidades de los tres actores del sistema: clientes, proveedores y administradores. Este análisis permitió definir con precisión los módulos que debía contemplar la plataforma y estableció la base de trazabilidad que guio tanto el desarrollo como la posterior validación, dando respuesta a la primera pregunta de investigación.

Respecto al segundo objetivo específico, el diseño de la arquitectura del sistema, el modelo de base de datos y la interfaz de usuario se formalizó mediante diagramas de casos de uso, de secuencia, de clases y de componentes. La arquitectura cliente-servidor en tres capas y el modelo documental sobre MongoDB fueron diseñados considerando separación de responsabilidades, control de acceso y posibilidad de crecimiento. Sin embargo, la escalabilidad y la seguridad no fueron verificadas mediante pruebas de carga ni auditorías

especializadas; por ello, estas características se presentan como decisiones de diseño y no como cualidades empíricamente garantizadas.

En cuanto al tercer objetivo específico, la implementación de los módulos funcionales con Node.js, Express, MongoDB y React.js permitió materializar la autenticación por roles, la verificación de proveedores, la publicación y búsqueda de anuncios y la gestión de solicitudes de servicio con propuesta de citas, e incorporó notificaciones automáticas mediante la WhatsApp Business Cloud API. El control de acceso basado en roles demostró particular solidez, pues los escenarios que probaron intentos de operación por parte de actores no autorizados retornaron de forma consistente el código HTTP 403, confirmando el correcto funcionamiento del middleware de autorización granular y respondiendo a la tercera pregunta de investigación.

Finalmente, en relación con el cuarto objetivo específico, la validación del sistema mediante 23 escenarios de prueba funcionales que verifican explícitamente los 21 requerimientos funcionales mediante trazabilidad directa o agrupada del SRS arrojó un resultado de 23 PASS y 0 FAIL, con lo que el 100 % de los casos de prueba ejecutados resultó satisfactorio sobre los requerimientos definidos. El despliegue de la plataforma en Railway, disponible en <https://servilink.up.railway.app> con integración continua desde el repositorio Git, garantizó su disponibilidad operativa en un entorno de producción en la nube, dando respuesta a la cuarta pregunta de investigación.

En conjunto, los resultados permiten concluir que ServiLink cumple el objetivo general como solución tecnológica local, funcional y verificable. La plataforma incorpora información de perfiles y un mecanismo de verificación documental que pueden contribuir a reducir la asimetría de información en la contratación de servicios independientes. No obstante, el efecto real sobre los riesgos de contratación y la informalidad deberá evaluarse posteriormente mediante un piloto en condiciones reales, indicadores de adopción y datos de uso.

4.8. Recomendaciones

Con base en lo aprendido a lo largo del desarrollo y en los resultados alcanzados, se plantean las recomendaciones que se detallan a continuación:

- A diferencia de la prueba de usabilidad cualitativa realizada con 30 participantes, se recomienda ejecutar una fase piloto en condiciones reales de operación, con un grupo de proveedores verificados y clientes durante un periodo definido, que permita medir indicadores de adopción, uso efectivo y satisfacción mediante instrumentos estructurados.
- Mantener la trazabilidad entre requerimientos, casos de uso y casos de prueba como práctica de aseguramiento de la calidad en las siguientes iteraciones, dado que fue determinante para verificar la cobertura funcional del sistema.
- Reforzar de forma continua la seguridad de la plataforma mediante la rotación periódica de las claves secretas almacenadas en variables de entorno, la

incorporación de limitación de tasa (rate limiting) en los endpoints públicos y la auditoría regular de las dependencias del proyecto.

- Implementar respaldos automáticos y periódicos de la base de datos, así como el monitoreo del entorno de producción, para garantizar la integridad de la información y la continuidad del servicio.
- Para futuros desarrolladores que continúen el proyecto, conservar la arquitectura por capas y la separación de responsabilidades adoptada, ya que facilita la incorporación de nuevos módulos sin comprometer la estabilidad del sistema.
- Realizar pruebas formales de rendimiento y carga (por ejemplo, con Apache JMeter o k6) que permitan verificar de manera empírica el cumplimiento de los requerimientos no funcionales de rendimiento, en particular RNF-01 (tiempo de respuesta de la API ≤ 2 s), antes de una liberación con mayor volumen de tráfico.

4.9. Trabajos futuros

El alcance del proyecto delimitó de manera deliberada un conjunto de funcionalidades que se plantean como líneas de trabajo futuro para la evolución de la plataforma:

- Integrar una pasarela de pagos en línea que permita gestionar las transacciones entre clientes y proveedores directamente en la plataforma, cerrando el ciclo de contratación de extremo a extremo.
- Incorporar un sistema de calificaciones y reseñas que fortalezca la confianza entre los usuarios y complemente el mecanismo de verificación de proveedores.
- Desarrollar un módulo de chat en tiempo real entre cliente y proveedor para facilitar la coordinación previa a la contratación del servicio.
- Añadir geolocalización con un mapa interactivo que permita a los clientes ubicar y filtrar proveedores por cercanía.
- Construir una aplicación móvil nativa para las plataformas iOS y Android que amplíe el alcance y la accesibilidad del servicio.
- Extender el panel de administración con un módulo de analítica que ofrezca indicadores de uso, demanda por categoría y desempeño de los proveedores, como apoyo a la toma de decisiones.

Referencias

- Agencia de Regulación y Control de las Telecomunicaciones. (2023). Boletín estadístico del sector de las telecomunicaciones 2023. <https://www.arcotel.gob.ec/boletines-estadisticos/>
- Alvear Yáñez, W. F. (2023). *Desarrollo de aplicación móvil multiplataforma para la contratación de profesionales de servicios generales* [Trabajo de titulación, Escuela Politécnica Nacional]. Repositorio Digital EPN. <https://bibdigital.epn.edu.ec/handle/15000/24316>
- Bajaña Vásquez, M. del C. (2021). *Diseño de una plataforma web basada en un modelo de negocio de intermediación en la distribución de productos y servicios* [Trabajo de titulación, Universidad Politécnica Salesiana]. Repositorio Institucional UPS. <https://dspace.ups.edu.ec/handle/123456789/21669>
- Espinoza Vásquez, D., & Barrera Penafiel, R. (2019). *Plan de negocio para la gestión en la contratación de freelancers relacionados al marketing digital mediante la implementación de una plataforma web* [Trabajo de titulación, Escuela Superior Politécnica del Litoral]. DSpace ESPOL. <https://www.dspace.espol.edu.ec/handle/123456789/47092>
- Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* [Tesis doctoral, University of California, Irvine]. <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación* (6.^a ed.). McGraw-Hill.

- Instituto Nacional de Estadística y Censos. (2022). Encuesta Nacional de Empleo, Desempleo y Subempleo (ENEMDU) anual 2022.
<https://www.ecuadorencifras.gob.ec/enemdu-anual-2022/>
- International Labour Organization. (2018). Women and men in the informal economy: A statistical picture (3rd ed.). <https://www.ilo.org/publications/women-and-men-informal-economy-statistical-picture-third-edition>
- International Telecommunication Union — Telecommunication Standardization Sector. (2018). *The international public telecommunication numbering plan* (ITU-T Recommendation E.164). <https://www.itu.int/rec/T-REC-E.164>
- Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)* (RFC 7519). Internet Engineering Task Force. <https://doi.org/10.17487/RFC7519>
- Meta for Developers. (2024). *WhatsApp Business Platform: Cloud API documentation*. Meta Platforms. <https://developers.facebook.com/docs/whatsapp/cloud-api>
- Meta Platforms. (2023). *React: A JavaScript library for building user interfaces*. <https://react.dev>
- Ministerio de Telecomunicaciones y de la Sociedad de la Información. (2022). Agenda de transformación digital del Ecuador 2022-2025.
<https://www.telecomunicaciones.gob.ec/wp-content/uploads/2022/08/Agenda-transformacion-digital-2022-2025.pdf>
- MongoDB Inc. (2024). *MongoDB manual: Documents and collections*.
<https://www.mongodb.com/docs/manual>
- Mongoose. (2024). *Mongoose: Elegant MongoDB object modeling for Node.js*.
<https://mongoosejs.com>

National Institute of Standards and Technology. (2017). *Digital identity guidelines* (Special Publication No. 800-63-3). U.S. Department of Commerce.

<https://doi.org/10.6028/NIST.SP.800-63-3>

OECD, International Labour Organization, & European Union. (2023). Handbook on measuring digital platform employment and work. OECD Publishing.

<https://doi.org/10.1787/0ddcac3b-en>

Open Web Application Security Project. (2024). *Password storage cheat sheet*. OWASP Foundation.

https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

OpenJS Foundation. (2024a). *Express: Fast, unopinionated, minimalist web framework for Node.js*. <https://expressjs.com>

OpenJS Foundation. (2024b). *Multer middleware documentation*. Express.js.

<https://expressjs.com/en/resources/middleware/multer.html>

OpenJS Foundation. (2024c). *Node.js documentation: The Node.js event loop*.

<https://nodejs.org/en/docs>

Parker, G., Van Alstyne, M., & Choudary, S. P. (2016). *Platform revolution: How networked markets are transforming the economy*. W. W. Norton & Company.

Railway. (2024). *Railway documentation: Deployments*. <https://docs.railway.com>

Romero Hernández, G. S. (2023). *Plataforma web para la gestión de trabajos freelance en Ecuador* [Trabajo de titulación, Universidad San Francisco de Quito]. Repositorio Digital USFQ. <https://repositorio.usfq.edu.ec/handle/23000/13936>

- Sandhu, R., Ferraiolo, D., & Kuhn, R. (2000). *The NIST model for role-based access control: Towards a unified standard*. Proceedings of the Fifth ACM Workshop on Role-Based Access Control, 47–63. <https://doi.org/10.1145/344287.344301>
- Schwaber, K., & Sutherland, J. (2020). *The Scrum guide*. Scrum.org. <https://scrumguides.org>
- Schwellnus, C., Geva, A., Pak, M., & Veiel, R. (2019). *Gig economy platforms: Boon or bane?* (OECD Economics Department Working Paper No. 1550). OECD. <https://doi.org/10.1787/fdb0570b-en>
- Secretaría Nacional de Planificación del Ecuador. (2021). Plan de creación de oportunidades 2021-2025. <https://www.planificacion.gob.ec/plan-de-creacion-de-oportunidades-2021-2025/>
- Tailwind Labs. (2024). *Tailwind CSS documentation: Utility-first fundamentals*. <https://tailwindcss.com/docs>
- Tanenbaum, A. S., & Van Steen, M. (2017). *Distributed systems: Principles and paradigms* (3rd ed.). Pearson.
- Vite. (2024). *Vite: Next generation frontend tooling*. <https://vite.dev>
- Workana. (2026). About Workana. <https://www.workana.com/en/about>
- Upwork. (2026). Identity verification: Frequently asked questions. Upwork Help. <https://support.upwork.com/hc/en-us/articles/34397755511955-Identity-verification-Frequently-asked-questions>
- Taskrabbitt. (2026). Taskrabbitt global terms of service. <https://support.taskrabbitt.com/hc/en-us/articles/46260465608603-Taskrabbitt-Global-Terms-of-Service>
- ServiLink. (2026). Informe de pruebas funcionales: Postman v11 y Swagger UI [Documento técnico del proyecto].

Herrera Gómez, R. A., & Quishpe Catagña, J. A. (2020). Desarrollo de sistema web y aplicación móvil para la contratación de servicios profesionales en el hogar [Trabajo de titulación, Escuela Politécnica Nacional]. Repositorio Digital EPN. <https://bibdigital.epn.edu.ec/handle/15000/21050>

Fiverr. (2026). Verifica tu identidad. Centro de Ayuda de Fiverr. <https://help.fiverr.com/hc/es/articles/13127850435345-Verifica-tu-identidad>

Anexos

Anexo A. Matriz completa de requerimientos funcionales

La presente matriz consolida los 21 requerimientos funcionales del sistema ServiLink, identificados durante la fase de análisis y documentados en la especificación de requerimientos de software. Se indica, para cada requerimiento, su identificador, nombre, módulo, actor responsable y prioridad.

Tabla 7

Matriz completa de requerimientos funcionales de ServiLink

ID	Nombre	Módulo	Actor	Métrica o criterio	Método de verificación	Evidencia/estado	
RF-01	Criterio definido en SRS	Revisión documental	No especificado	Alta		Criterio definido en SRS	Revisión documental
No especificado	Registro Proveedor (directo)	Autenticación	Criterio definido en SRS	Revisión documental	No especificado		
Criterio definido en SRS	Revisión documental	No especificado	Todos	Alta	Criterio definido en SRS	Revisión documental	No especificado
RF-04	Cierre de sesión	Criterio definido en SRS	Revisión documental	No especificado			Criterio definido en SRS
Revisión documental	No especificado	Verificación	Proveedor	Criterio definido en SRS	Revisión documental	No especificado	
RF-06	Criterio definido en SRS	Revisión documental	No especificado	Alta		Criterio definido en SRS	Revisión documental
No especificado	WhatsApp: resultado verificación	Verificación	Criterio definido en SRS	Revisión documental	No especificado		
Criterio definido en SRS	Revisión documental	No especificado	Cliente/Dual	Alta	Criterio definido en SRS	Revisión documental	No especificado
RF-09	Perfil público proveedor	Criterio definido en SRS	Revisión documental	No especificado			Criterio definido en SRS

ID	Nombre	Módulo	Actor	Métrica o criterio	Método de verificación	Evidencia/estado	
Revisión documental	No especificado	Solicitudes	Cliente/Dual	Criterio definido en SRS	Revisión documental	No especificado	
RF-11	Criterio definido en SRS	Revisión documental	No especificado	Alta		Criterio definido en SRS	Revisión documental
No especificado	WhatsApp: eventos de cita	Solicitudes	Criterio definido en SRS	Revisión documental	No especificado		
Criterio definido en SRS	Revisión documental	No especificado	Cliente/Dual	Alta	Criterio definido en SRS	Revisión documental	No especificado
RF-14	Rol dual: Proveedor como Cliente	Rol dual	Proveedor	Alta			
RF-15	Upgrade Cliente → Proveedor	Rol dual	Cliente	Alta			
RF-16	Revisión upgrade por Admin	Rol dual	Administrador	Alta			
RF-17	Panel de administración	Administración	Administrador	Media			
RF-18	Gestión de usuarios	Administración	Administrador	Media			
RF-19	Publicación de anuncios	Anuncios	Proveedor	Alta			
RF-20	Reporte de anuncios	Anuncios	Todos (autent.)	Alta			
RF-21	Gestión anuncios reportados	Anuncios	Administrador	Alta			

Anexo B. Matriz completa de requerimientos no funcionales

La presente matriz consolida los 19 requerimientos no funcionales del sistema, agrupados por categoría de calidad (rendimiento, seguridad, usabilidad, mantenibilidad, escalabilidad, disponibilidad y despliegue). Cada requerimiento incluye su identificador, categoría, descripción y prioridad.

Tabla 8***Matriz completa de requerimientos no funcionales de ServiLink***

ID	Categoría	Descripción	Prioridad
RNF-01	Rendimiento	El tiempo de respuesta de la API para operaciones de lectura (búsqueda, listado) no debe superar los 2 segundos bajo condiciones normales.	Alta
RNF-02	Rendimiento	El sistema debe soportar al menos 50 usuarios concurrentes sin degradación perceptible del rendimiento.	Media
RNF-03	Rendimiento	Las imágenes de perfil y documentos cargados deben procesarse y almacenarse en menos de 5 segundos.	Media
RNF-04	Seguridad	Toda comunicación entre cliente y servidor debe realizarse sobre HTTPS con certificado TLS válido.	Alta
RNF-05	Seguridad	Las contraseñas se almacenarán con hash bcrypt, con factor de coste mínimo de 10 rondas; nunca en texto plano.	Alta
RNF-06	Seguridad	Los endpoints de la API estarán protegidos mediante JWT; los de administración verificarán adicionalmente el rol de administrador.	Alta
RNF-07	Seguridad	El sistema implementará protección contra inyección NoSQL en todas las consultas a MongoDB.	Alta
RNF-08	Seguridad	Los documentos de identidad se almacenarán en una ruta no pública, accesible solo por el administrador autenticado.	Alta
RNF-09	Seguridad	El token de acceso de WhatsApp Business API se almacenará como variable de entorno y nunca se expondrá en el repositorio.	Alta
RNF-10	Usabilidad	La interfaz debe ser responsiva y funcionar en dispositivos móviles desde 360 px de ancho.	Alta
RNF-11	Usabilidad	Los formularios mostrarán mensajes de error claros y descriptivos junto al campo correspondiente.	Media
RNF-12	Usabilidad	El flujo de registro no debe requerir más de 3 pasos para completarse.	Media
RNF-13	Usabilidad	La interfaz estará completamente en español, adaptada al contexto ecuatoriano.	Alta
RNF-14	Usabilidad	El dashboard del usuario con rol dual mostrará claramente las secciones "Como cliente" y "Como proveedor", diferenciadas visualmente.	Alta
RNF-15	Mantenibilidad	El código fuente estará organizado por responsabilidad (controllers, routes, models, middlewares, services) siguiendo el patrón MVC.	Media
RNF-16	Mantenibilidad	Las variables sensibles (base de datos, JWT secret, token y Phone Number ID de WhatsApp) se gestionarán con un archivo .env y nunca se subirán al repositorio.	Alta

RNF-17	Escalabilidad	La arquitectura permitirá agregar nuevas categorías de servicio y nuevos eventos de notificación WhatsApp sin modificar la lógica central.	Media
RNF-18	Disponibilidad	El sistema debe estar disponible en línea con una disponibilidad objetivo del 99 % mensual.	Media
RNF-19	Despliegue	El sistema se desplegará en Railway con un pipeline de despliegue continuo desde Git.	Alta

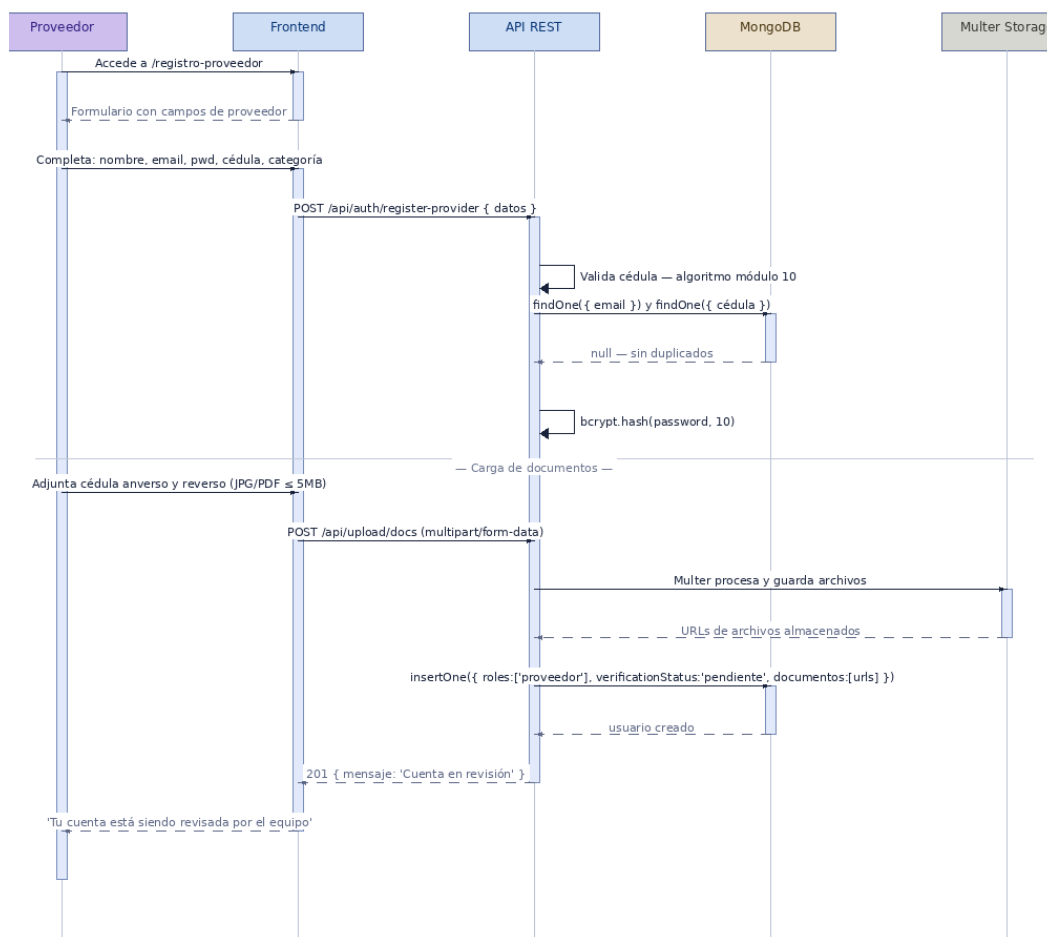
Anexo C. Diagramas de secuencia complementarios

El Capítulo III presentó cuatro diagramas de secuencia representativos de los flujos principales del sistema. A continuación se incluyen los seis diagramas de secuencia complementarios que completan la documentación de los flujos temporales entre los participantes del sistema, con indicación de los requerimientos funcionales que cada uno cubre.

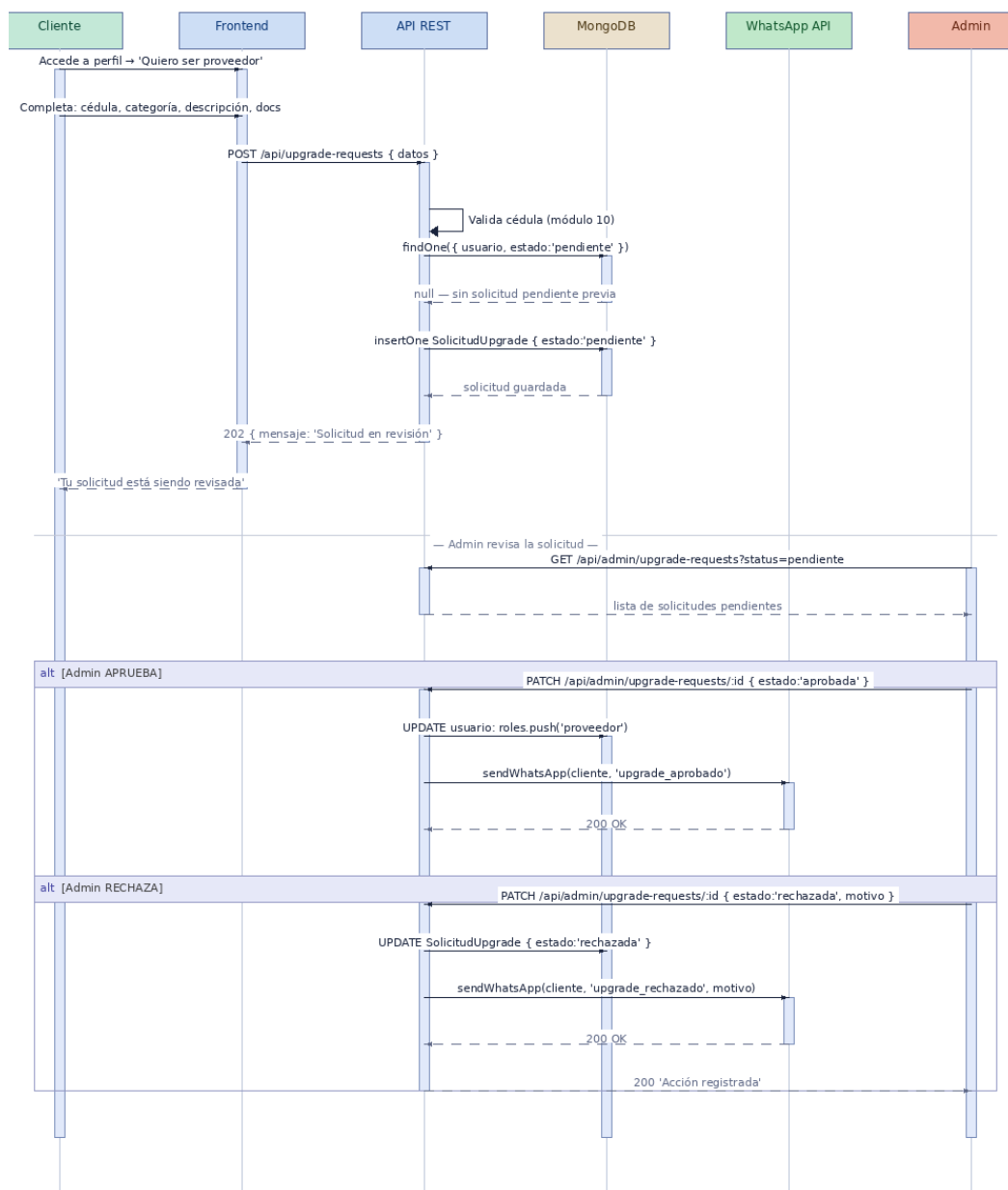
La Figura 42 presenta el diagrama de secuencia de registro de cliente (RF-01). Muestra el flujo de creación de una cuenta de cliente, desde el envío del formulario hasta la persistencia del usuario y la respuesta del sistema.

Figura 42
Diagrama de Secuencia — Registro de Cliente (RF-01)

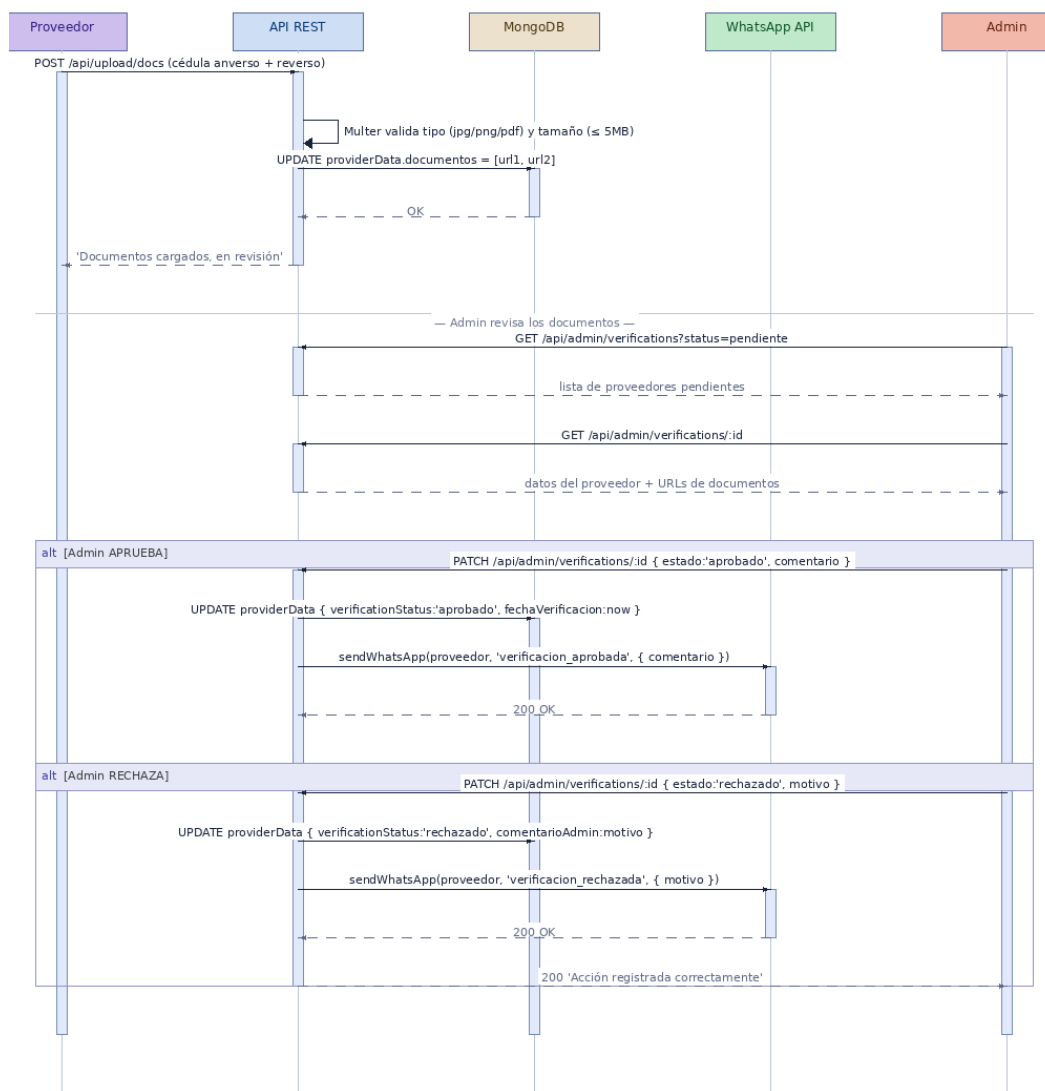
Seq 2 — Registro de Proveedor



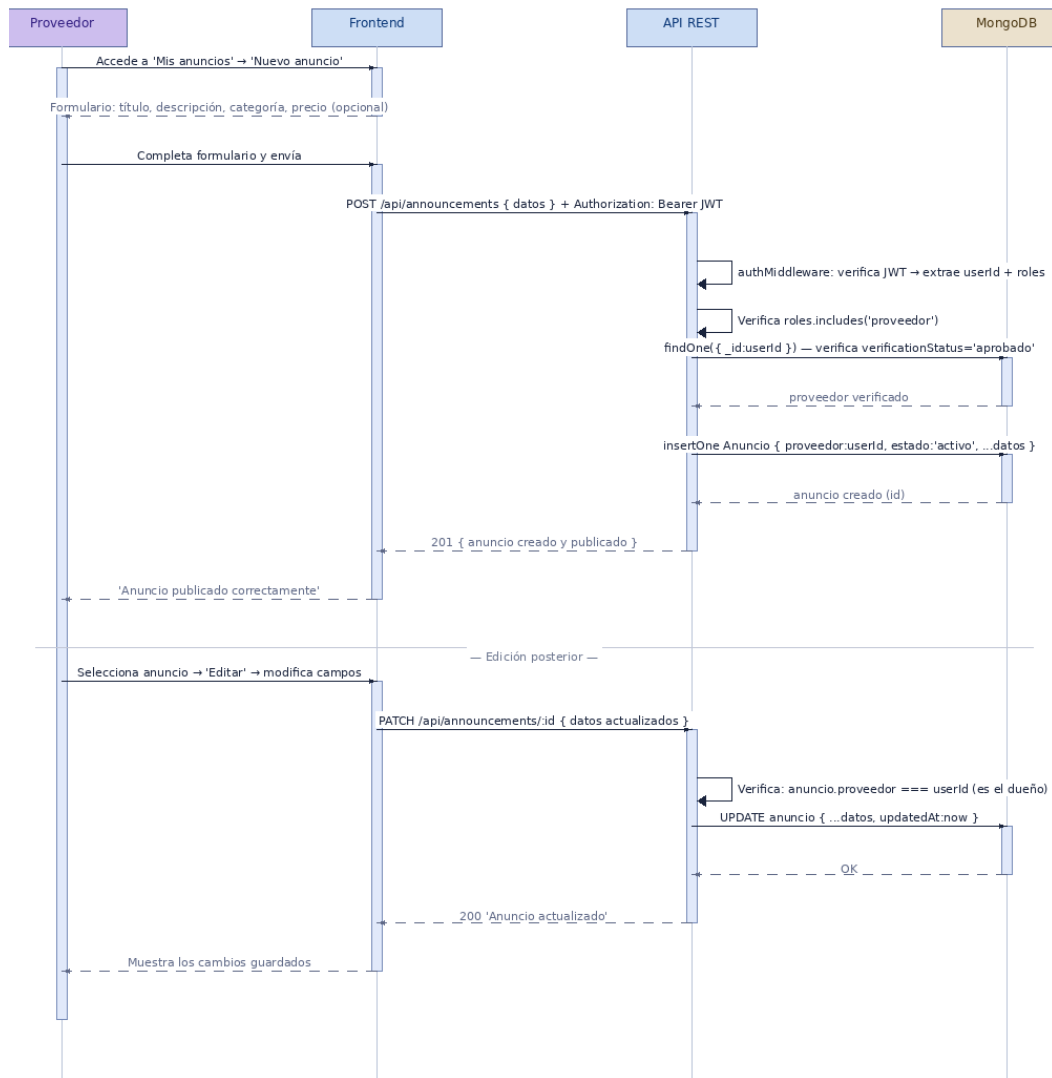
La Figura 43 presenta el diagrama de secuencia de inicio de sesión con jwt (RF-03). Representa la autenticación de un usuario: validación de credenciales, generación del token JWT y su devolución al cliente.

Figura 43**Diagrama de Secuencia — Inicio de Sesión con JWT (RF-03)****Seq 4 — Upgrade de Rol: Cliente → Proveedor**

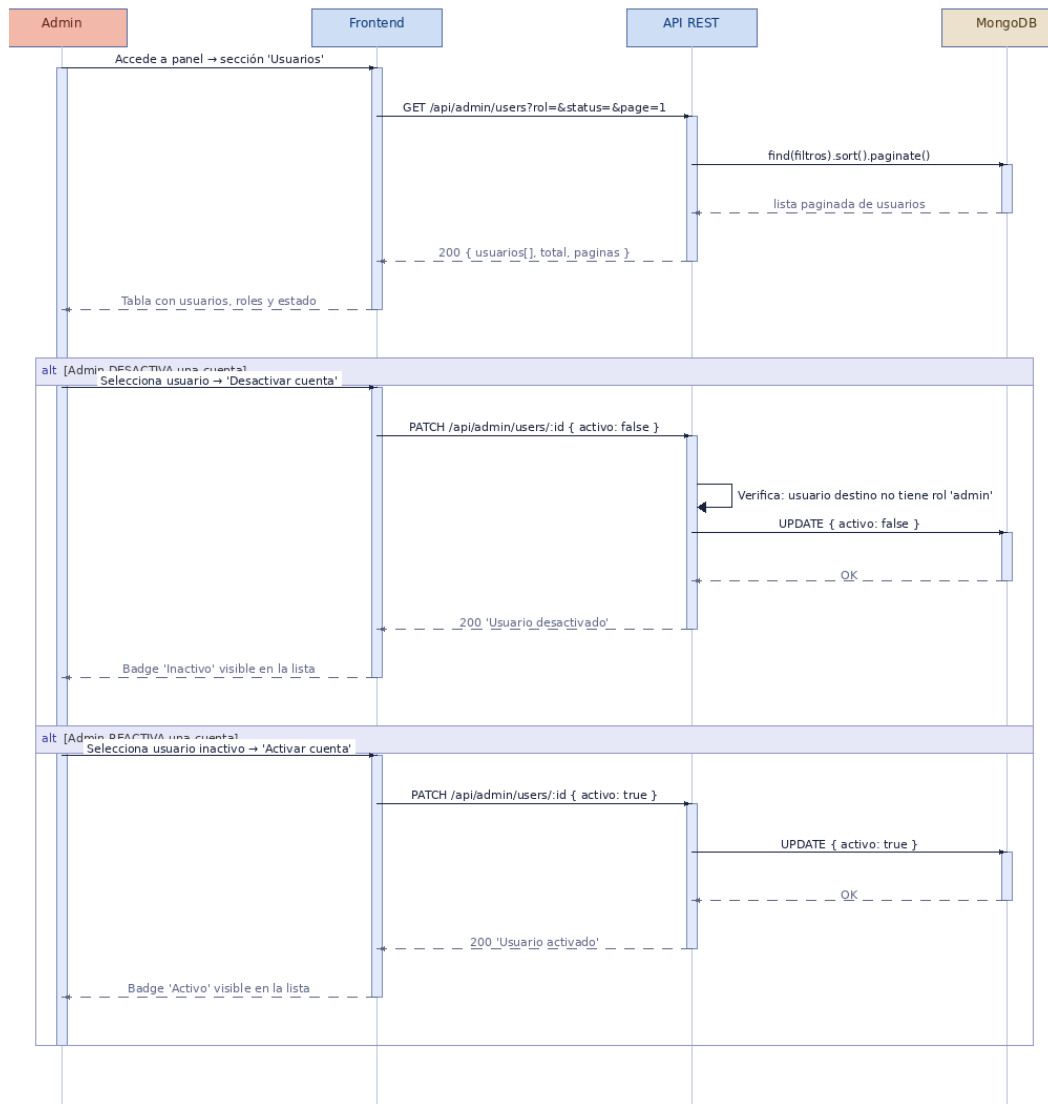
La Figura 44 presenta el diagrama de secuencia de reagendamiento de cita (RF-11, RF-12). Detalla la propuesta de una nueva fecha por parte del proveedor, la respuesta del cliente y la notificación automática por WhatsApp.

Figura 44*Diagrama de Secuencia — Reagendamiento de Cita (RF-11, RF-12)***Seq 7 — Verificación de Proveedor**

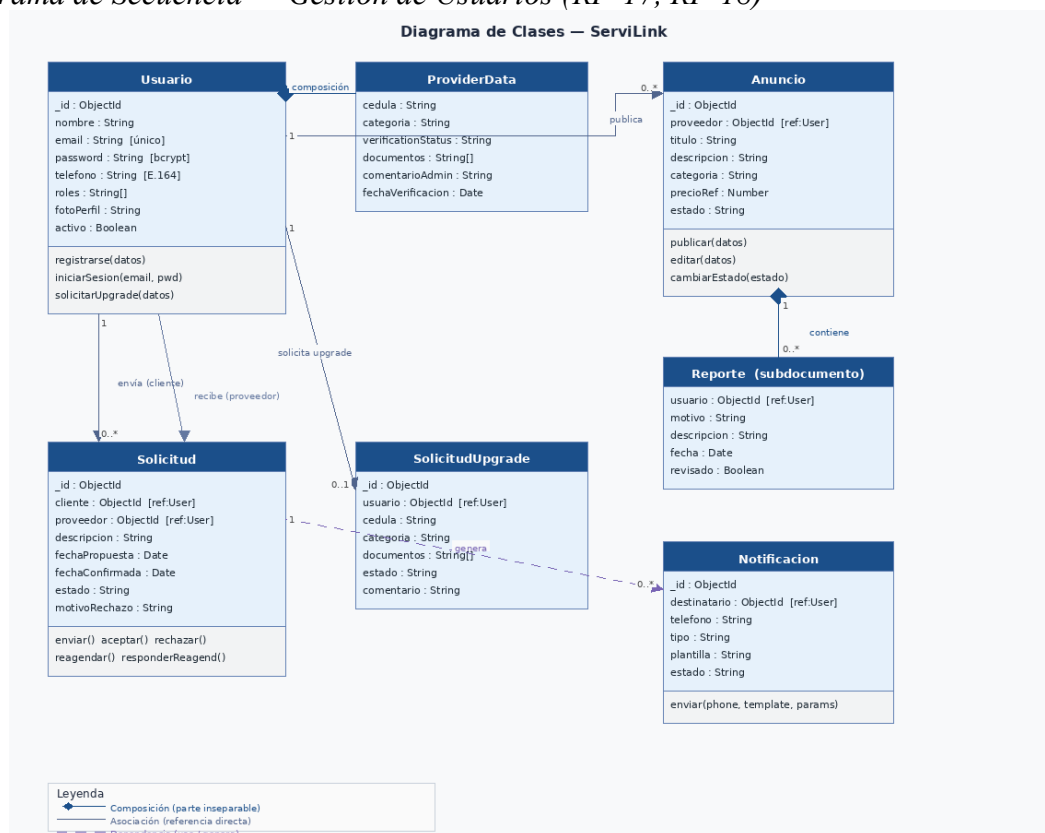
La Figura 45 presenta el diagrama de secuencia de reporte y gestión de anuncio (RF-20, RF-21). Describe el reporte de un anuncio por un usuario y la posterior revisión y acción del administrador sobre dicho anuncio.

Figura 45*Diagrama de Secuencia — Reporte y Gestión de Anuncio (RF-20, RF-21)***Seq 9 — Publicación de Anuncio por Proveedor**

La Figura 46 presenta el diagrama de secuencia de publicación de anuncio (RF-19). Muestra la creación de un anuncio por un proveedor verificado, incluyendo la validación de su estado antes de publicarlo.

Figura 46*Diagrama de Secuencia — Publicación de Anuncio (RF-19)***Seq 10 — Gestión de Usuarios por Administrador**

La Figura 47 presenta el diagrama de secuencia de gestión de usuarios (RF-17, RF-18). Representa la consulta y administración de usuarios por parte del administrador desde el panel de control.

Figura 47*Diagrama de Secuencia — Gestión de Usuarios (RF-17, RF-18)*

Anexo D. Matriz de trazabilidad entre requerimientos, casos de uso, diagramas y pruebas

La matriz relaciona cada requerimiento funcional con el flujo, diagrama o evidencia de prueba que permite comprobar su implementación. Cuando un requerimiento no posee un caso independiente, se verifica como parte de un flujo integrado o mediante inspección del módulo correspondiente.

RF	Flujo/módulo	Caso o diagrama	Evidencia
RF-01	Registro/autenticación	CU autenticación	Prueba registro cliente
RF-02	Registro proveedor	Secuencia registro proveedor	Tres escenarios de cédula
RF-03	Autenticación	CU autenticación	Login JWT
RF-04	Autenticación	CU autenticación	Logout
RF-05	Verificación	CU verificar proveedor	Control de roles y carga
RF-06	Verificación	Secuencia verificación	Listado de pendientes
RF-07	Verificación/WhatsApp	Secuencia verificación	Aprobar/rechazar y log
RF-08	Búsqueda	CU búsqueda y perfil	Listado y detalle público

RF-09	Perfil proveedor	CU búsqueda y perfil	Inspección de respuesta pública
RF-10	Solicitudes	CU contratar servicio	Creación de solicitud
RF-11	Solicitudes	CU gestionar solicitud	Aceptar/rechazar/reagendar
RF-12	Notificaciones	Secuencia contratación	Logs de eventos de cita
RF-13	Seguimiento	CU gestionar solicitud	Respuesta al reagendamiento
RF-14	Rol dual	CU rol dual	Restricción de autosolicitud
RF-15	Upgrade	CU solicitar upgrade	Flujo de solicitud de upgrade
RF-16	Upgrade admin	Secuencia upgrade	Revisión administrativa
RF-17	Administración	CU panel admin	Dashboard
RF-18	Usuarios	CU panel admin	Listado y filtros
RF-19	Anuncios	CU gestión de anuncios	Crear anuncio con controles
RF-20	Reportes	Secuencia reporte	Reportar anuncio
RF-21	Moderación	Secuencia reporte	Ver anuncios reportados

Anexo E. Autorización de publicación en el Repositorio

**AUTORIZACIÓN DE PUBLICACIÓN EN EL
REPOSITORIO INSTITUCIONAL**

Juan Carlos Guamán Tituaña portador(a) de la cédula de ciudadanía N° 0107190977. En calidad de autor/a y titular de los derechos patrimoniales del trabajo de titulación **“Diseño e implementación de una plataforma web para la oferta y contratación de servicios independientes con verificación de proveedores”** de conformidad a lo establecido en el artículo 114 Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación, reconozco a favor de la Universidad Católica de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos y no comerciales. Autorizo además a la Universidad Católica de Cuenca, para que realice la publicación de este trabajo de titulación en el Repositorio Institucional de conformidad a lo dispuesto en el artículo 144 de la Ley Orgánica de Educación Superior.

Cuenca, 31 de julio de 2026

F: 

Juan Carlos Guamán Tituaña

C.I. 0107190977