Acta de Constitución del Proyecto

# Información del proyecto

## Datos

| Empresa / Organización | TeLoCambio (Proyecto Académico - Duoc UC) |
| --- | --- |
| Proyecto | Plataforma Web “TeLoCambio” |
| Fecha de preparación | 27-08-2025 |
| Cliente | Administración de condominio/edificio/junta de vecinos |
| Patrocinador principal | Fernando Gonzalo Herrera Francesconi |
| Gerente de proyecto | Octavio Quezada |
| Versión | 1.0 |

# Propósito y justificación del proyecto

| El proyecto TeLoCambio busca desarrollar una plataforma que facilite el intercambio de bienes y servicios entre usuarios de una misma comunidad, utilizando una arquitectura moderna desacoplada.  El proyecto responde a dos necesidades clave: la acumulación de objetos en buen estado que ya no se utilizan y la falta de herramientas confiables y seguras para coordinar trueques o servicios entre residentes cercanos , generando así valor social en contextos de baja liquidez o colaboración comunitaria.  **Elevator Pitch** "Imagina que tienes un libro que ya no leerás o un electrodoméstico que no usas. En lugar de botarlo, ¿no sería genial poder cambiarlo por algo que sí necesitas?”  Hoy, el consumo desmedido y la generación de desechos son un problema creciente. Por eso hemos creado **TeLoCambio**: una plataforma web que fomenta la economía circular permitiendo a los vecinos de una comunidad **truequear objetos** de manera fácil y segura. |
| --- |

# Descripción del proyecto y entregables

| El objetivo general es diseñar, construir, desplegar y documentar una solución integral y funcional de trueque de bienes y servicios para comunidades cerradas. La plataforma operará bajo un  El sistema incluirá un proceso de registro controlado por código de comunidad y aprobación de un administrador , módulos completos para la publicación y búsqueda de bienes y servicios, doble confirmación de intercambios y un sistema de calificación. Adicionalmente, se desarrollará un panel de administración para la gestión de usuarios y la moderación de contenido.  Los 3 focos técnicos principales de la aplicación son los siguientes:   * Plataforma web SPA (Single Page Application) desarrollada con React + TypeScript. * Backend seguro y escalable usando Django REST Framework (Python). * Base de datos PostgreSQL.   **Entregables Principales:**  **Código fuente completo** (frontend, backend e infraestructura como código).  **Base de datos** y procedimientos documentados de respaldo/restauración.  **Manual de usuario** (para residentes) y **Manual de administración** (para administradores de comunidad).  **Guía de despliegue y operación** detallando la contenerización y variables de entorno.  **Plan y evidencias de pruebas** (unitarias, de integración y de aceptación).  **Presentación final** con una demostración funcional de la plataforma.  **Product Backlog** a detalle según sprint e historias de usuario realizadas.  Registro de reuniones y tareas realizadas según día.  **Mockups** |
| --- |

# Objetivos

| **Objetivo** | **Indicador de éxito** |
| --- | --- |
| **Alcance** | |
| **1. Desarrollar una plataforma funcional que cubra todo el flujo de bienes y servicios** | La plataforma permite: registro, publicación, búsqueda, doble confirmación y calificación. Validado con pruebas funcionales del 100% de los casos de uso esperados. |
| **2. Implementar control de acceso por roles y tenencia multi-comunidad** | El sistema permite acceso diferenciado entre residentes y administradores, y aísla los datos por comunidad (multi-tenant), comprobado con pruebas de penetración y tests de acceso. |
| **3. Incorporar panel de administración con capacidad de auditoría básica** | El administrador puede ver métricas clave, gestionar usuarios/publicaciones y revisar logs de actividad. Validado en demo funcional con test de aceptación. |
| **Cronograma (Tiempo)** | |
| **4. Cumplir el cronograma académico propuesto sin retrasos críticos** | El proyecto se completa dentro del calendario oficial, entregando los hitos principales (diseño, MVP, pruebas, despliegue, documentación) en las fechas definidas. |
| **5. Completar el desarrollo funcional del MVP en un máximo de 6 semanas** | Las funcionalidades mínimas viables están listas, integradas y probadas dentro del plazo estipulado. Validado en revisión de avance con profesor guía. |
| **6. Realizar entregas incrementales cada 2 semanas** | Cada sprint entrega funcionalidades funcionales verificables. Validado por el historial de commits y despliegues en staging. |
| **Costo** | |
| **7. Ejecutar el proyecto sin gastos económicos directos** | Todos los recursos utilizados (infraestructura, herramientas, librerías) son gratuitos o de licencia académica. Confirmado por documentación técnica y listado de herramientas. |
| **8. Minimizar el uso de recursos computacionales para optimizar la carga en entornos gratuitos** | El consumo promedio de CPU/RAM del backend se mantiene dentro de los límites gratuitos de Heroku/Azure en staging. Validado con métricas en consola de despliegue. |
| **9. Maximizar la reutilización de componentes de software open-source** | Al menos 3 paquetes o librerías externas utilizadas de forma efectiva para reducir el tiempo de desarrollo. Validado en análisis de dependencias del repositorio. |
| **Calidad** | |
| **10. Asegurar alto rendimiento en la experiencia de usuario** | El 95% de las vistas principales se cargan en ≤ 2 segundos. Medido con herramientas como Lighthouse o DevTools. |
| **11. Garantizar seguridad básica: HTTPS, autenticación y protección contra accesos no autorizados** | Toda la comunicación es cifrada (HTTPS), el acceso está autenticado y cada endpoint tiene validaciones de permisos. Validado en pruebas de seguridad. |
| **12. Lograr una cobertura de pruebas automatizadas ≥ 80%** | Ejecutar tests unitarios e integración en backend con coverage ≥ 80%. Validado en CI/CD pipeline. |
| **Otros** | |
| **13. Lograr una interfaz intuitiva para usuarios sin experiencia técnica** | Al menos 85% de usuarios piloto navega exitosamente por la plataforma sin ayuda, en pruebas de usabilidad. |
| **14. Asegurar accesibilidad desde dispositivos móviles y escritorio sin errores** | Funcionalidad completa confirmada en navegadores desktop y móviles. Validado en pruebas multi-dispositivo. |
| **15. Mantener una documentación técnica clara y actualizada** | Repositorio con README, diagramas y manual de despliegue técnico actualizado y validado por revisor externo. |

# 

# Exclusiones( Fuera de Alcance en esta versión)

1. **Procesamiento de pagos/escrow y logística externa.**

La plataforma no implementará pagos electrónicos, gestión de dinero en custodia (escrow) ni servicios de logística de despacho. Esto se excluye porque el propósito central del proyecto es facilitar el *trueque* y el intercambio comunitario, no generar un sistema de comercio electrónico completo. Además, incorporar medios de pago requiere integraciones financieras, normativas legales y costos adicionales que exceden el alcance académico y el tiempo disponible.

1. **Integraciones con mapas públicos o validación geográfica por proximidad.**

Aunque la ubicación puede ser un factor importante en plataformas de intercambio, la integración con servicios de mapas (ej. Google Maps) y validaciones de proximidad se excluye por ahora debido a la complejidad técnica y las dependencias de APIs externas que podrían generar costos. En esta versión inicial, la coordinación de lugar se delega a los usuarios vía contacto externo (ej. WhatsApp).

1. **Verificación de identidad avanzada (RUT/biometría).**

La validación de identidad mediante documentos oficiales o biometría no será implementada en esta etapa, ya que requiere infraestructura adicional, manejo de datos sensibles y cumplimiento de normativas legales de protección de datos. Para la versión actual, la autenticación básica con correo y contraseña será suficiente para fines académicos y de prueba.

1. **Sistema de mensajería interno y negociaciones.**

No se desarrollará un sistema de mensajería privada dentro de la plataforma. En su lugar, las negociaciones se derivarán a aplicaciones externas como WhatsApp mediante *deeplink*. Se excluye porque implementar mensajería implica un alto nivel de complejidad (almacenamiento de mensajes, seguridad, notificaciones en tiempo real) y podría desviar recursos del objetivo principal: validar el flujo de intercambio.

# 

# 

# 

# Riesgos iniciales de alto nivel

| **A. Riesgos técnicos y funcionales de la aplicación**   * **Onboarding lento de usuarios:** El proceso de incorporación puede ser lento debido a requisitos de aprobación para unirse a la comunidad.   + **Mitigación:** Facilitar el ingreso mediante códigos comunitarios preaprobados. Crear plantillas de invitación para la administración y automatizar aprobaciones comunes. * **Masa crítica insuficiente:** La plataforma necesita un número mínimo de usuarios para ser útil.   + **Mitigación:** Coordinar campañas de difusión interna con la administración (carteles, QR). * **Conductas inadecuadas:** Posible mal uso de la plataforma (spam, publicaciones inadecuadas, conflictos entre usuarios).   + **Mitigación:** Incorporar funciones de bloqueo y moderación para administradores. Hacer visible el sistema de reputación (calificaciones). Incluir términos de uso aceptados al registrarse. * **Vulnerabilidad de seguridad en la API:** Riesgo de que usuarios malintencionados realicen acciones no autorizadas.   + **Mitigación:** Aplicar control de acceso basado en roles utilizando el principio de mínimo privilegio en todos los endpoints. Usar las herramientas de seguridad integradas de Django (CSRF, hashing de contraseñas). Registrar auditoría de acciones críticas. * **Complejidad en la gestión de agenda y doble confirmación:** Módulos sensibles que requieren coordinación entre usuarios podrían fallar en lógica o usabilidad.   + **Mitigación:** Validar los flujos mediante pruebas de usuario. Mantener lógica clara y registrar cada estado de confirmación.   **B. Riesgos del equipo de trabajo y gestión del proyecto**   * **Desviación en tiempos y alcance:** Riesgo de no completar todas las funcionalidades planeadas dentro del plazo.   + **Mitigación:** Aplicar priorización basada en valor (MoSCoW o similar) en cada sprint. Eliminar o postergar funcionalidades "nice-to-have" si el tiempo es limitado. * **Dificultad de coordinación entre frontend y backend:** Al ser un equipo desacoplado, puede haber desfase en entregas o malentendidos técnicos.   + **Mitigación:** Definir claramente contratos de API desde etapas tempranas. Mantener reuniones breves de sincronización y documentar endpoints. * **Escalabilidad limitada:** La plataforma podría enfrentar problemas si aumenta significativamente la carga de usuarios.   + **Mitigación:** Desplegar la infraestructura con contenedores Docker para facilitar la escalabilidad. Usar una base de datos robusta (PostgreSQL). Planificar pruebas de carga en fase final. * **Problemas de despliegue en entornos gratuitos**: Heroku, Netlify o similares pueden presentar límites de uso o disponibilidad.   + **Mitigación:** Realizar pruebas de despliegue anticipadas. Tener alternativas de hosting (ej. Azure, BlueHosting). Minimizar dependencias externas no críticas. |
| --- |

# Cronograma de hitos principales

| **Hito** | **Fecha tope** |
| --- | --- |
| 1. Definición y Planificación | Semana 4 |
| 2. Construcción I (Core Funcional) | Semana 8 |
| 3. Construcción II (Funciones Avanzadas) | Semana 13 |
| 4. Calidad y Operación | Semana 15 |
| 5. Cierre y Entrega Final | Semana 16 |

# 

| Hito | Fecha estimada | Objetivo / Entregable esperado |
| --- | --- | --- |
| Sprint 0 – Planificación y Setup | Semana 1 a 3 | Definición del Product Backlog inicial, configuración del repositorio, estructura de carpetas, asignación de roles, y diseño de wireframes. |
| Sprint 1 – MVP Básico (Core funcional) | Semana 4 a 7 | Funcionalidades mínimas: Registro, login, publicación de objetos, visualización. |
| Sprint 2 – Funcionalidades avanzadas | Semana 8 a 12 | Agenda de intercambios, reputación, panel de administración, doble confirmación. |
| Sprint 3 – Refactor y pruebas | Semana 13 a 14 | Mejora de código, seguridad, pruebas funcionales, validación con usuarios reales. |
| Sprint 4 – Documentación y entrega final | Semana 15 a 16 | Manual de usuario, presentación final, versión de producción estable. |

# 

# Presupuesto inicial asignado

| No se asigna un presupuesto monetario. El proyecto se ejecutará con los recursos de infraestructura (hosting, software) disponibles en el contexto académico. |
| --- |

# Lista de Interesados (stakeholders)

| **Nombre** | **Cargo** | **Departamento / División** |
| --- | --- | --- |
| Gonzalo Herrera Francesconi | DUOC UC Administración de Comunidad | DUOC UC |
| Administración de Comunidad | Cliente (B2B) | Condominio/Edificio/Junta de Vecinos |
| Residentes (+18) | Usuarios Finales (B2C) | Condominio/Edificio/Junta de Vecinos |
| Rodrigo Barra  Paul López  Octavio Quezada | Integrante del Equipo | Equipo de Proyecto |

# Niveles de autoridad

| **Área de autoridad** | **Descripción del nivel de autoridad** |
| --- | --- |
| Decisiones de personal  (Staffing) | El equipo es auto-organizado. La asignación de tareas se decide en las reuniones de planificación de sprint. |
| Gestión de presupuesto  y de sus variaciones | No aplica, ya que no hay un presupuesto monetario asignado. |
| Decisiones técnicas | Las decisiones técnicas son tomadas por el equipo de desarrollo en consenso, siguiendo la arquitectura definida. |
| Resolución de conflictos | Los conflictos se abordan primero dentro del equipo. Si no se resuelven, se escalan al Product Owner (Docente Guía). |
| Ruta de escalamiento y  limitaciones de autoridad | Equipo → Product Owner (Docente Guía). |

# Personal y recursos preasignados

| **Recurso** | **Departamento / División** |
| --- | --- |
| Rodrigo Barra | Developer (Backend, seguridad, autenticación) |
| Paul López | Developer (Base de datos, pruebas, integración) |
| Octavio Quezada | Developer (Frontend, React, UI/UX) |
| Recursos Tecnológicos | React, Django, PostgreSQL, Docker, GitHub, Trello/Azure DevOps |

# Aprobaciones

| **Patrocinador** | **Fecha** | **Firma** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |