Universidad Nacional de La Matanza

Cátedra Trabajo Final

**Año 2025**

**Proyecto:**

**“Smart Contractors”**

**Número de Equipo:**

**Integrantes Equipo de Proyecto**

|  |  |  |
| --- | --- | --- |
| DNI | Nombre | E-Mail |

**ÍNDICE**

[1. Modelo de Negocio 2](#_Toc634671934)

[1.1 Business Model Canvas 3](#_Toc1392983802)

[1.2 Explicación de los nueve módulos 3](#_Toc1058977358)

[1.3 Oferta - Cuadro de competidores 4](#_Toc2099863691)

[1.4 Oferta - Productos complementarios 4](#_Toc2073645463)

[1.5 Análisis económico-financiero 4](#_Toc996556895)

[1.5.1 Parámetro del periodo 4](#_Toc313421722)

[1.5.2 Resultado del periodo (ganancia) 5](#_Toc175516816)

[1.5.3 Inversión inicial requerida (antes de empezar) 5](#_Toc474687911)

[1.5.4 Plan de uso de la ganancia 5](#_Toc943386109)

[2.Definición de alcance 7](#_Toc1833158634)

[2.1 En / Fuera de alcance 7](#_Toc1886039649)

[2.2 Visual story Mapping 7](#_Toc1826525638)

[2.3 Product Backlog 7](#_Toc1438766158)

[2.4 Criterios de aceptación 7](#_Toc352525299)

[3. Interesados y Equipo 9](#_Toc1081682052)

[3.1 Matriz de interesados 9](#_Toc107335642)

[3.2 Equipo de proyecto 9](#_Toc1253511383)

[4. Release planning - Plan de versiones 10](#_Toc851323201)

[4.1 Estimación Story Points 10](#_Toc971357052)

[4.2 Plan de versiones 10](#_Toc1810702873)

[5. Experiencia de Usuario 11](#_Toc1384466053)

[5.1 Prototipo de navegación 11](#_Toc500957838)

[5.2 Wireframes Mockups 11](#_Toc311940596)

[6. Arquitectura de Software 11](#_Toc1210553694)

[6.1 Diagrama de Arquitectura 12](#_Toc255260774)

[6.2 Frameworks / Componentes utilizados 12](#_Toc95341568)

[6.3 Infraestructura tecnológica 12](#_Toc183031231)

[6.4 Otros diagramas según corresponda 13](#_Toc2133285964)

[6.5 Estándar de codificación 13](#_Toc2010698986)

[7. Plan de pruebas 13](#_Toc1456612042)

[7.1 Diseño y ejecución de escenario de prueba 14](#_Toc856881456)

[7.2 Seguimiento de fallas 14](#_Toc262038245)

[8. Manual Instalación 14](#_Toc861609820)

[9. Hoja de Ruta 15](#_Toc1002449841)

[Anexos I. Retrospectiva 16](#_Toc1390985082)

# 1. Modelo de Negocio

## 1.1 Business Model Canvas 1.2 Explicación de los nueve módulos

1.2.1 Segmento de mercado

El proyecto esta principalmente para freelancers y creadores de contenido digital (Diseño, fotografía, redacción, edición de video, etc) que buscan un entorno seguro para ofrecer sus servicios, y a clientes particulares o empresas que desean contratar este tipo de servicios digitales de forma confiable, transparente y global.

1.2.2 Propuesta de valor

Ofrecemos **seguridad y transparencia** en la contratación de servicios digitales mediante el uso de **contratos inteligentes en blockchain**, que automatizan los acuerdos y eliminan riesgos de incumplimiento. A través de un sistema de **escrow**, los pagos quedan retenidos en custodia hasta que el cliente aprueba el trabajo, garantizando así confianza para ambas partes

También, se reduce la dependencia de intermediarios, se agilizan los pagos con criptomonedas y se brinda un flujo de trabajo **auditable y automatizado mediante logs en la blockchain**. Un aspecto diferencial de la plataforma es la posibilidad de realizar pagos y cobros con **stablecoins (USDC, con paridad 1:1 con el dólar estadounidense)**, lo que permite a los freelancers acceder a ingresos en dólares digitales de manera segura. Esto representa un gran valor agregado en países como Argentina, donde el acceso a dólares es limitado y los pagos internacionales suelen ser costosos y complicados. De esta forma, la plataforma no solo ofrece **confianza tecnológica**, sino también una **ventaja económica real** para los usuarios que buscan cobrar o pagar de manera más estable y conveniente

1.2.3 Canales

Los principales canales a través de los cuales los clientes y freelancers podrán descubrir y acceder a la plataforma incluyen, en primer lugar, una **landing page oficial**, que funcionará como puerta de entrada para explicar la propuesta de valor, registrar usuarios y redirigirlos a la aplicación web y móvil, conectadas a la tesnet por MetaMask. Ademas, la difusion se apoya en redes sociales profesionales (Como LinkedIn, Behance e Instagram), donde los diseñadores graficos y otros creadores de contenido son de promocionar sus trabajos y captar clientes. La captación de usuarios se reforzará a través de la participación en **eventos académicos y meetups tecnológicos**, donde se presentará la plataforma a estudiantes, desarrolladores y potenciales clientes

1.2.4 Relaciones con los clientes

Seria en un **onboarding simple mediante Metamask**, que interactúa con la blockchain sin molestias. Se busca **confianza y autonomía**, ya que los contratos inteligentes automatizan gran parte del proceso de contratación y pago. Además de que hay una gran transparencia mediante **event logs** de la blockchain

Para consultas rápidas y frecuentes, la plataforma contara con una sección de FAQs. Para problemas más específicos o de complejidad, puede haber canales de contacto directo como correos electrónicos, formulario de soporte

También la comunidad de usuarios podrá interactuar y recibir ayuda en grupos oficiales, fomentando la colaboración entre cliente y freelancer

1.2.5 Fuentes de ingresos

* Comisión por transaccion en los contratos exitosos
* Publicidad no intrusiva dentro de la plataforma
* Integraciones estratégicas con wallets y exchanges

1.2.6 Recursos clave

**Tecnológico**:

* Contratos inteligentes en Solidity, con testnet Sepolia
* Wallet Metamask, para gestion de indentidad y transaccion
* Token USDC de testnet para pagos
* Frontend en React Next.Js, con Web3.js
* Backend y servicios en la nube
* Repositorio en Github

**Organización**:

* Gestión de tareas y flujo de trabajo con Trello
* Procesos internos para coordinación y soporte a clientes
* Identidad de marca para generar confianza y posicionar la plataforma

**Humano**:

* Equipo de desarrolladores especializados en blockchain y contratos inteligentes
* Expertos en seguridad informática
* Profesionales en diseño de producción, UX y marketing
* Personal de soporte al cliente

**Inversionistas:**

* Red de inversionistas que aporten capital inicial para el desarrollo y crecimiento de la plataforma
* Posibles asociaciones estratégicas con fondos de inversión en tecnología y blockchain
* Aliados financieros interesados en la innovación en pagos digitales y economía freelance

**Cultura:**

* Filosofía empresarial basada en la **transparencia, confianza y descentralización**
* Promoción de la **innovación continua** y adopción de nuevas tecnologías
* Fomento de un entorno de trabajo **colaborativo, inclusivo y orientado a la comunidad**
* Compromiso con la **educación y capacitación constante** del equipo y los usuarios

1.2.7 Actividades clave

* Desarrollo y pruebas de contratos con blockchain
* Escrow Pattern en los contratos (Fondos en custodia)
* Integración de frontend con MetaMask y Blockchain
* Definición del State Machine

1. Cliente crea trabajo y deposita fondos (Open)
2. FreeLancer acepta (In progress)
3. FreeLancer entrega (Pending review)
4. Cliente aprueba y libera fondos (Completed)

* Testing de casos completos, de cliente y freelancers
* Mantenimiento de infraestructura y repositorios (Gestión)
* Provisión de canales de ayuda para resolver dudas y acompañar al usuario
* Gestión de seguridad (Auditorias, control de accesos, prevención de fraudes y normas legales)
* Marketing para promoción de la plataforma en diferentes redes
* Coordinación de tareas y planificados
* Gestión financiera, incluyendo administración de ingresos, egresos y relación con inversionistas.
* Fortalecimiento de la cultura y comunidad, tutoriales y meetups que consoliden la identidad de la plataforma

1.2.8 Socios clave - Proveedores

* Proveedores de testnet y la nube AWS
* Expertos en contratos inteligentes
* Asesores legales y ciberseguridad
* Comunidades de Web3 y Blockchain como fuente de apoyo

1.2.9 Estructura de costos

* Desarrollo de contratos inteligentes y auditorias de seguridad
* Desarrollo web y mobile
* Infraestructura (nube AWS, repositorios, integración continua y monitoreos)
* Soporte de usuario y capacitación (FAQs, atención personalizada y formación continua para equipo de soporte)
* Asesoría legal para cumplimiento regulatorio
* Costo del personal/salarios (Desarrolladores, diseñadores, soporte, marketing y administración)
* Mantenimiento tecnológico (Programación y pruebas, auditorias)
* Marketing y publicidad (Campañas en redes sociales, eventos, generación contenido digital)
* Costos operativos (Licencias de software, herramientas de gestión, gastos administrativos y contables)
* Costos financieros: gestión contable, impuestos, pagos a proveedores y comisiones de servicios externos
* Inversión en innovación y estrategia: actividades de I+D (Investigación y desarrollo), relación con inversionistas y desarrollo de nuevas funcionalidades para la plataforma

## 1.3 Oferta - Cuadro de competidores

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre del producto | Sitio Web | Fortalezas | Debilidades |
| Freelancer | https://www.freelancer.com/ | - Amplia base global: +50 millones de usuarios.  - Alta velocidad de respuesta (propuestas en <1 minuto).  - Modelo sólido: escrow, reputación, soporte 24/7, app móvil.  - Diversidad de categorías (más de 1800). | - Competencia muy alta (freelancers de todo el mundo).  - Bajos precios iniciales, presión por competir en tarifas.  - Posible pérdida de proyectos por contacto fuera de la plataforma.  - Costos de membresía y comisiones pueden ser altos. |
| Workana | https://www.workana.com/ | - Plataforma líder en Latinoamérica.  - Enfocada en hispanohablantes y cultura local  - Sistema de reputación con niveles (Pro, Hero, etc.)  - Herramientas para fijar precios: calculadora de tarifa, simuladores. | - Comisiones altas para freelancers (hasta 20%).  - Planes pagos para aumentar visibilidad.  - Saturación de algunas categorías populares.  - Interfaz limitada en funcionalidades para empresas grandes. |
| Laburas | https://laburas.com/ | - Plataforma argentina, con enfoque local.  - Procesos transparentes y soporte humano cercano.  - Sistema de escrow, pagos en pesos y USD.  - Servicio "Laburas Empresas" para contratación rápida.  - Comunidad pequeña, pero segmentada en talento nacional. | - Escala pequeña y baja visibilidad fuera de Argentina.  - Menor volumen de proyectos y clientes.  - Limitada variedad de categorías técnicas.  - Sitio en crecimiento, con funcionalidades aún en desarrollo |

## 1.4 Oferta - Productos complementarios

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre del producto | Sitio Web | Fortalezas | Debilidades |
| MetaMask | https://metamask.io/ | - Acceso seguro y descentralizado.  - Permite interacción directa con contratos inteligentes.  - Amplia adopción. - Control Total sobre los Fondos | - Curva de aprendizaje.  - Posibles problemas de seguridad para usuarios inexpertos. - Falta de Soporte de Recuperación de cuenta. |
| Circle (USDC - Testnet/Mainnet) | https://www.circle.com/ | - Stablecoin Respalda por el dólar. - Regulación y Auditoría - Amplia Compatibilidad y Conversión | - Posibles Restricciones Regulatorias  - Dependencia de una Entidad Centralizada  - Comisiones de Gas Altas en Ethereum |
| AWS | https://aws.amazon.com/ | - Escalabilidad.  - Seguridad y Fiabilidad.  - Servicios Específicos para Blockchain | - Costos Elevados  - Complejidad en la Gestión. - Dependencia de un Solo Proveedor |
| LinkedIn | https://www.linkedin.com/ | - Comunicación Directa y Rápida.  - Popularidad Global.  Fácil Integración con la Plataforma | - Saturación de comunidades, ruido de comunicación.  - Poco Control sobre el Entorno. |

## 1.5 Análisis económico-financiero

### 1.5.1 Parámetro del periodo

|  |  |
| --- | --- |
| Concepto | Valor |
| Periodo (ej.: anual) | Mensual (proyección a 12 meses) |
| Unidades por vender (U) | 1.000 usuarios activos promedio al mes (se proyecta que cada usuario realice 2 transacciones/mes = 2.000 transacciones mensuales) |
| Precio unitario (P) | Comisión del 5% por transacción. Suponiendo ticket promedio de USD 100, ingreso promedio de USD 5 por transacción |
| Costo variable unitario (CV) | USD 1 por transacción (gastos de red blockchain, integraciones y procesamiento) |
| Costos fijos del periodo (CF) | USD 8.000 mensuales (infraestructura en la nube, sueldos mínimos del equipo base, marketing inicial, soporte y administración) |

### 1.5.2 Resultado del periodo (ganancia)

|  |  |  |
| --- | --- | --- |
| Concepto | Formula | Resultado |
| Ingresos (I) | U x P | USD 10.000 |
| Costo variable total (CVT) | U x CV | USD 2.000 |
| Costo total del periodo | CVT + CF | USD 10.000 |
| Ganancia del periodo (R) | I − CVT − CF | USD 0 |
|  |  |  |

### 1.5.3 Inversión inicial requerida (antes de empezar)

|  |  |  |
| --- | --- | --- |
| Ítem (setup/licencias/equipamiento/domino, etc.) | Monto unitario | Monto total |
| Registro de dominio | USD 15 | USD 15 |
| Hosting inicial en AWS (plan básico + almacenamiento + CI/CD setup) | USD 50 | USD 600 |
| Licencias de software de gestión (Trello, herramientas de diseño UX) | USD 12 | USD 144 |
| Identidad visual y branding inicial (logo, diseño base UI/UX) | USD 400 | USD 400 |
| Desarrollo y Auditoría inicial de contratos inteligentes (testnet Sepolia) | USD 1400 | USD 1400 |
| Setup de repositorios privados | USD 48 | USD 48 |
| Marketing digital inicial (campañas en LinkedIn, Behance, IG) | USD 300 | USD 3600 |
| Asesoría legal/regulatoria | USD 700 | USD 700 |
| Equipamiento mínimo adicional | USD 3000 | USD 3000 |
| Total de inversión inicial | USD 5925 | USD 9907 |

*La inversión inicial se paga con fondos propios por USD 9907*

### 1.5.4 Plan de uso de la ganancia

|  |  |  |  |
| --- | --- | --- | --- |
| Destino | % de la ganancia | Monto estimado | Detalle |
| Reinvención en el proyecto | 40% | USD 4000 | Desarrollar nuevas funcionalidades, mejoras tecnicas, mantenimiento, auditorias de seguridad |
| Marketing | 20% | USD 2000 | Campaña en redes sociales, participacion de eventos, workshops |
| Fondo de reserva | 20% | USD 2000 | Fondo para cubrir imprevistos, meses negativos o caídas de ingresos |
| Equipo (Bono/premios) | 15% | USD 1500 | Incentivos y bonos por desempeño para colaboradores claves |
| Otros (Asesorías, partnerships) | 5% | USD 500 | Consultoras especializadas y alianzas estratégicas con empresas con blockchain |
| Total | 100% | USD 10000 | Distribución sobre las ganancias en USD |

# 2.Definición de alcance

## 2.1 En / Fuera de alcance

|  |  |  |  |
| --- | --- | --- | --- |
| Épica | En alcance | Fuera de alcance | Nota |
| Registro e identidad del usuario | Conexión con wallet (ej. MetaMask), sin registro tradicional | Registro con email y contraseña, validación por correo | Se usará la wallet como identidad única |
| Crear publicación de servicio digital | Crear una publicación indicando el servicio ofrecido o buscado (ej: "necesito edición de video") | Validaciones avanzadas o moderación de contenido | Frontend básico con formulario y guardado on-chain o en base de datos persistiendo información básica |
| Búsqueda y visualización de publicaciones | Ver publicaciones disponibles en un listado | Sistema de ranking, filtros avanzados o búsquedas con AI |  |
| Crear contrato entre partes | Contrato inteligente básico entre dos wallets que define términos simples (servicio, monto, tiempo estimado) | Contratos multiparte, cláusulas condicionales complejas | Contrato en Solidity desplegado en testnet (Sepolia) |
| Firma del contrato | Ambas partes deben firmar desde sus wallets para aceptar | Delegación de firmas, firmas fuera de la app | Firma en blockchain usando MetaMask |
| Seguimiento de estado del contrato | Ver si el contrato está: propuesto, aceptado, en ejecución, finalizado | Automatización del cambio de estado por actividad externa | Cambio de estado manual o al firmar |
| Historial de contratos del usuario | Ver contratos pasados en los que participó | Historial completo con filtros, exportación a PDF | Frontend con datos en lista |
| Manejo de tokens (testnet) | Transacción con tokens de testnet (ej: ETH Sepolia), pero token usdc | Integración con stablecoins reales o pagos fiat | Se usará ETH de testnet, sin riesgos financieros reales |
| Interfaz en React integrada con MetaMask | Aplicación en React que permite conectar la wallet y operar | Aplicación móvil, integración con múltiples wallets | Solo versión web (desktop preferentemente) |
| Despliegue en blockchain de pruebas | Deploy en Sepolia u otra testnet compatible | Deploy en mainnet | Asegurar que todos los contratos estén bien probados antes del deploy |
| Seguridad del contrato | Validación de entradas básicas (e.g. que las wallets no sean iguales, datos no vacíos) | Auditoría formal de seguridad | Contrato simple y documentado |
| Documentación del proyecto | Documentar el código, decisiones técnicas, flujos del usuario y arquitectura | Manual de usuario detallado, documentación en varios idiomas | Incluir README, diagramas simples y comentarios en el código |

## 2.2 Visual story Mapping



## 2.3 Product Backlog

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID Historia de usuario** | **Como** | **Quiero** | **Para** | **Prioridad** | **Estatus** |
| **US01** | Cliente | Conectar mi wallet a la red de prueba (Sepolia) | Poder operar en la plataforma | Alta | Pendiente |
| **US02** | Cliente | Generar un nuevo trabajo indicando monto y detalles | Contratar a un trabajador | Alta | Pendiente |
| **US03** | Cliente | Aprobar al contrato el movimiento de mis USDC | Garantizar que el pago quede reservado | Alta | Pendiente |
| **US04** | Cliente | Depositar el monto en garantía en el contrato al crear el trabajo | Asegurar que el pago quede retenido | Alta | Pendiente |
| **US05** | Cliente | Ver el trabajo en estado CREADO | Confirmar que la publicación se registró correctamente | Alta | Pendiente |
| **US06** | Trabajador | Conectar mi wallet | Poder interactuar con la plataforma | Alta | Pendiente |
| **US07** | Trabajador | Visualizar la oferta de trabajos | Seleccionar los que me interesen | Alta | Pendiente |
| **US08** | Trabajador | Aceptar un trabajo disponible | Comenzar a realizarlo | Alta | Pendiente |
| **US09** | Trabajador | Ver el trabajo en estado EN PROGRESO | Saber que puedo comenzar a trabajar | Alta | Pendiente |
| **US10** | Trabajador | Enviar el trabajo terminado | Confirmar entrega | Alta | Pendiente |
| **US11** | Trabajador | Ver el trabajo en estado ENTREGADO | Confirmar que la entrega se registró | Alta | Pendiente |
| **US12** | Cliente | Revisar la entrega | Validar que el trabajo cumple con lo solicitado | Alta | Pendiente |
| **US13** | Cliente | Aprobar el trabajo | Autorizar la liberación de fondos | Alta | Pendiente |
| **US14** | Trabajador | Recibir la liberación de fondos en mi wallet | Cobrar de forma segura | Alta | Pendiente |
| **US15** | Cliente y Trabajador | Ver el trabajo en estado COMPLETO | Confirmar que el proceso finalizó correctamente | Alta | Pendiente |
| **US16** | Cliente | Revisar mi saldo de USDC de testnet | Saber si tengo fondos para publicar un trabajo | Media | Pendiente |
| **US17** | Trabajador | Rechazar un trabajo | Mantener organizada mi lista de trabajos | Media | Pendiente |
| **US18** | Trabajador | Ver el trabajo en estado RECHAZADO | Confirmar que fue rechazado | Media | Pendiente |
| **US19** | Cliente | Expire automáticamente si el freelancer no cumple el plazo | Para recuperar mis fondos | Alta | Pendiente |

## 2.4 Criterios de aceptación

## Historia de Usuario: #US02

*Como CLIENTE quiero CREAR UN CONTRATO INTELIGENTE para PODER CONTRATAR UN FREELANCER DE FORMA SEGURA*

|  |  |  |  |
| --- | --- | --- | --- |
| **N° Criterio de aceptación** | **Dado...** | **Cuando...** | **Entonces...** |
| 1 | que el cliente está conectado a su wallet MetaMask en la red, y está en la pantalla “Crear contrato” con saldo suficiente en USDC | ingresa una descripción del trabajo, un monto válido en USDC, un plazo de entrega en días y confirma la transacción en MetaMask | se despliega un contrato inteligente en la blockchain y se muestra el mensaje “Contrato creado exitosamente” |
| 2 | que el cliente está autenticado, pero no tiene saldo suficiente | intenta confirmar la transacción | el sistema rechaza la creación del contrato y muestra el mensaje sobre la insuficiencia de fondos |

Historia de Usuario: #US08

*Como FREELANCER quiero ACEPTAR UN CONTRATO para EMPEZAR A TRABAJAR*

|  |  |  |  |
| --- | --- | --- | --- |
| **N° Criterio de aceptación** | **Dado...** | **Cuando...** | **Entonces...** |
| 1 | que el freelancer está conectado a su wallet MetaMask y el contrato está en estado “Creado” | el freelancer haga clic en “Aceptar contrato” y confirme la transacción en MetaMask | el contrato cambiará su estado a *InProgress* y se genera un log en la blockchain y se muestra un mensaje de “Contrato aceptado” |
| 2 | que el contrato ya está en estado distinto de “Creado” | el freelancer intenta aceptarlo | el sistema bloquea la acción y muestra el mensaje “No está disponible para aceptar” |
| 3 | que el freelancer no está conectado a MetaMask | intenta aceptar un contrato | el sistema rechaza la acción y muestra el mensaje “Debe conectar su wallet para aceptar contratos” |

Historia de Usuario: #US10

*Como FREELANCER quiero ENVIAR EL TRABAJO FINAL para QUE EL CLIENTE LO REVISE*

|  |  |  |  |
| --- | --- | --- | --- |
| **N° Criterio de aceptación** | **Dado...** | **Cuando...** | **Entonces...** |
| 1 | que el contrato está en estado *InProgress* y el freelancer está conectado a su walle | el freelancer cargue la entrega en formato válido (PDF, DOCX, PNG, JPG, MP4 o ZIP hasta 100 MB) y confirme la transacción en MetaMask | el contrato pasará a estado *Submitted* y se registrará un event log en blockchain y se muestra el mensaje “Entrega enviada” |
| 2 | que el contrato no está en estado InProgress | el freelancer intenta enviar una entrega | el sistema rechaza la acción y muestra el mensaje “No se puede enviar entrega en este estado del contrato” |
| 3 | que el freelancer carga un archivo con formato no permitido | presiona “Enviar” | el sistema rechaza la entrega y muestra “Formato no admitido, cargue un archivo válido” |

Historia de Usuario: #US13

*Como CLIENTE quiero APROBAR EL TRABAJO para LIBERAR LOS FONDOS AL FREELANCER*

|  |  |  |  |
| --- | --- | --- | --- |
| **N° Criterio de aceptación** | **Dado...** | **Cuando...** | **Entonces...** |
| 1 | Que el contrato está en estado *Submitted* | el cliente apruebe la entrega en formato válido y confirme la transacción en MetaMask | el contrato pasa a completo y libera los fondos al freelancer y el cliente ve el mensaje ‘Entrega aprobada’, los fondos se liberan |
| 2 | que el contrato está en estado distinto de Submitted | el cliente intenta aprobarlo | el sistema rechaza la acción y muestra el mensaje “El contrato no está listo para aprobación” |

Historia de Usuario: # US19

*Como CLIENTE quiero QUE EL CONTRATO EXPIRE AUTOMATICAMENTE SI EL FREELANCER NO CUMPLE EL TIEMPO ACORDADO para RECUPERAR MIS FONDOS*

|  |  |  |  |
| --- | --- | --- | --- |
| **N° Criterio de aceptación** | **Dado...** | **Cuando...** | **Entonces...** |
| 1 | que el contrato tiene un plazo definido en días | el plazo se cumpla sin que el freelancer haya entregado el trabajo | el contrato pasará automáticamente a expirado y se registra un log en la blockchain y el cliente visualiza el mensaje ‘Contrato expirado, fondos devueltos’ |
| 2 | que el contrato está expirado | el sistema ejecute la función de expiración | los fondos en escrow serán devueltos al cliente, se muestra el mensaje “Contrato expirado, fondos devueltos” y quedará registrado un log en blockchain |

# 3. Interesados y Equipo

## 3.1 Matriz de interesados

|  |  |  |
| --- | --- | --- |
| Nombre Interesado | Rol | Contacto |
| UNLAM | Instituto educativo | www.unlam.edu.ar |
| Materia TP Final | Profesora y supervisora del TP final | UNLAM – Departamento de ING y EFC |
| Equipo del Proyecto | Desarrolladores de la aplicación | Trello / Teams |
| Usuarios (Clientes y Freelancers) | Futuros usuarios de la plataforma en pruebas | Contacto indirecto por simulaciones y testing |
| Inversores / Aliados (Futuro) | Posibles socios financieros o técnicos para escalar el producto | Contacto futuro por networking |

## 3.2 Equipo de proyecto

|  |  |
| --- | --- |
| Rol de Scrum | Nombre y apellido |
| Product Owner / Documentación | Leandro Alonso |
| Desarrollador backend / Integraciones | Arami Aquino |
| QA / Tester | Nahuel Batilicio |
| Scrum master / Arquitectura | Emiliano Castillo |
| Desarrollador frontend / Diseño UX/UI | Tomas Vázquez |

# 4. Release planning - Plan de versiones

## 4.1 Estimación Story Points

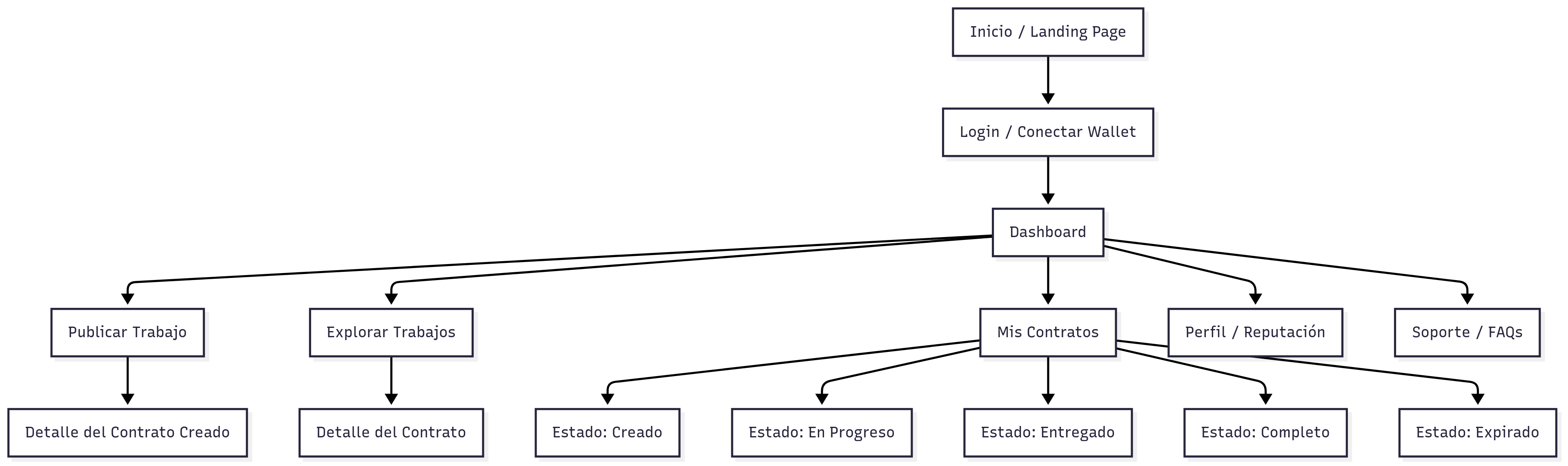
|  |  |  |
| --- | --- | --- |
| **ID HU** | **Historia de Usuario** | **Estimación** |
| US01 | Conectar wallet a Sepolia Testnet | 3 |
| US02 | Generar nuevo trabajo (monto, tiempo, descripción) | 8 |
| US03 | Aprobación de contrato y congelar USDC | 5 |
| US04 | Depositar monto al Smart Contract | 5 |
| US05 | Ver el trabajo en estado CREADO | 2 |
| US06 | Trabajador conecta wallet | 3 |
| US07 | Visualizar oferta de trabajos | 5 |
| US08 | Aceptar trabajo | 5 |
| US09 | Ver trabajo en estado EN PROGRESO | 2 |
| US10 | Enviar trabajo terminado (definir tipo de archivo permitido) | 8 |
| US11 | Ver trabajo en estado ENTREGADO | 2 |
| US12 | Cliente revisa la entrega | 5 |
| US13 | Aprobar trabajo | 5 |
| US14 | Recibir liberación de fondos | 3 |
| US15 | Ver trabajo en estado COMPLETO | 2 |
| US16 | Revisar saldo de USDC testnet | 3 |
| US17 | Rechazar trabajo | 3 |
| US18 | Ver trabajo en estado RECHAZADO | 2 |
| US19 | Expiración automática por incumplir plazos | 8 |

## 4.2 Plan de versiones

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Versión** | **Sprint** | **Inicio** | **Fin** | **Entrega comprometida (HU)** | **Notas adicionales** |
| **V1** | Sprint 1 | 17/09/2025 | 30/09/2025 | US01, US02, US03, US04, US05, US16 (conectar wallet cliente, crear trabajo, aprobar movimiento USDC, depositar fondos, ver trabajo creado, revisar saldo) | Se prioriza el flujo mínimo viable del cliente para crear contratos y reservar fondos. **Backend** enfocado en contratos inteligentes + integración MetaMask. |
| **V2** | Sprint 2 | 01/10/2025 | 14/10/2025 | US06, US07, US08, US09, US10, US11, US17, US18 (conectar wallet trabajador, visualizar y aceptar/rechazar trabajos, ver estados, enviar entrega) | Se habilita el flujo del trabajador. **Frontend** inicial conectado al backend. Incluye validaciones de estado. |
| **V3** | Sprint 3 | 15/10/2025 | 29/10/2025 | US12, US13, US14, US15, US05, US06, US07 (revisión de entrega, aprobación, liberación de fondos, contrato completo y expiración automática | Cierre del ciclo completo. Se agrega lógica de expiración. Pruebas integradas end-to-end. |

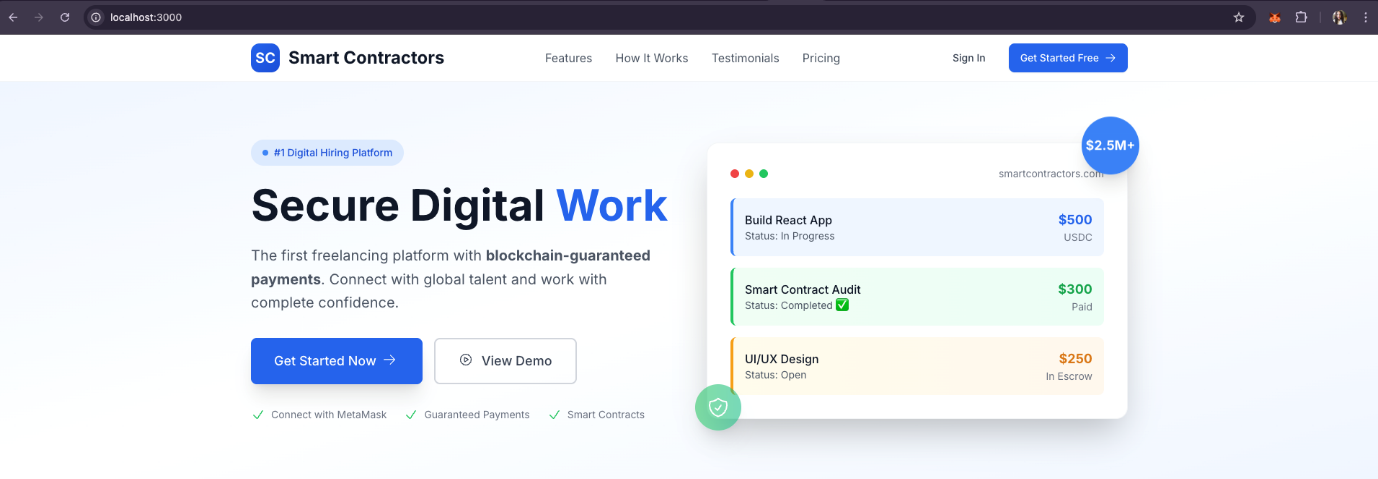
# 5. Experiencia de Usuario

## 5.1 Prototipo de navegación

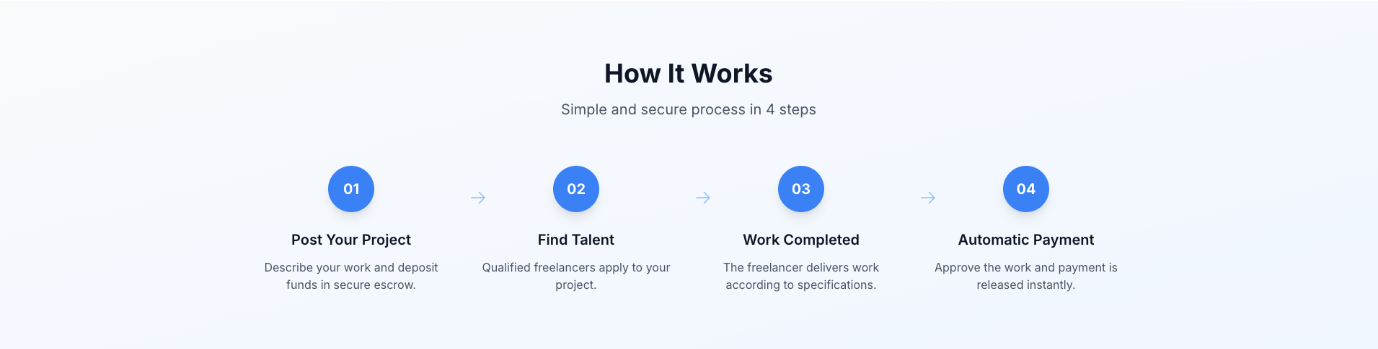


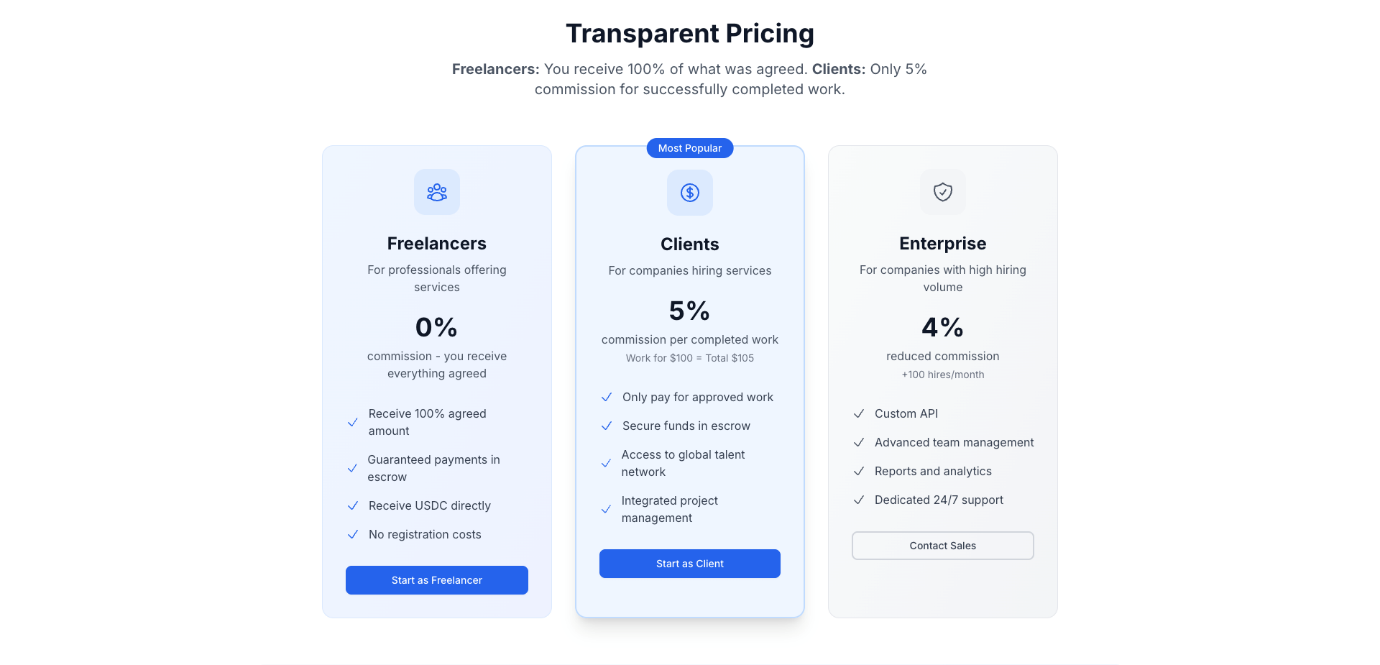
## 5.2 Wireframes Mockups

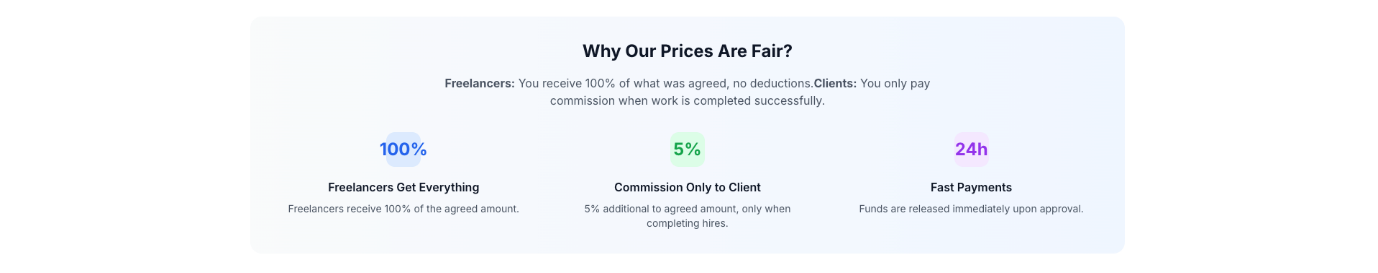
Landing Page

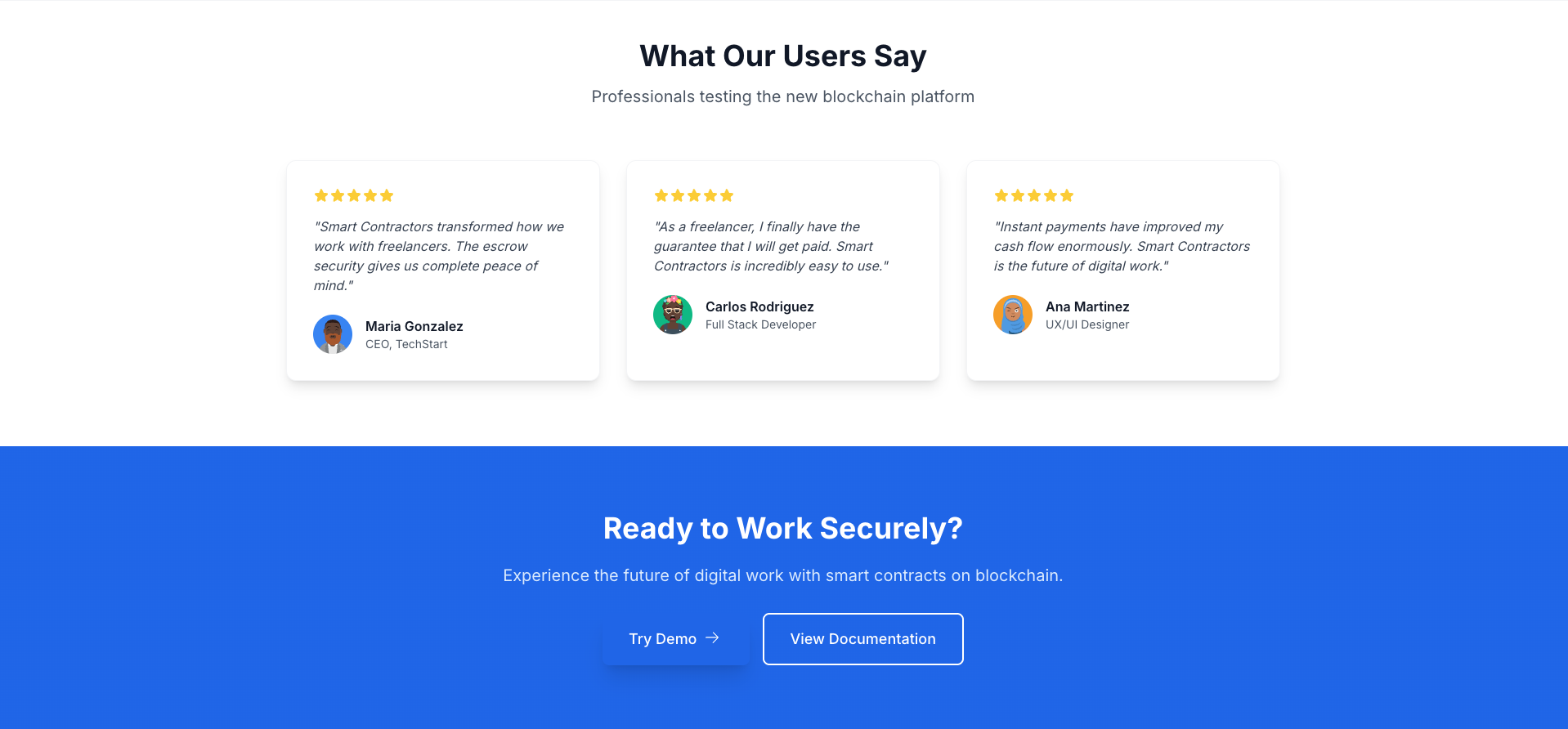


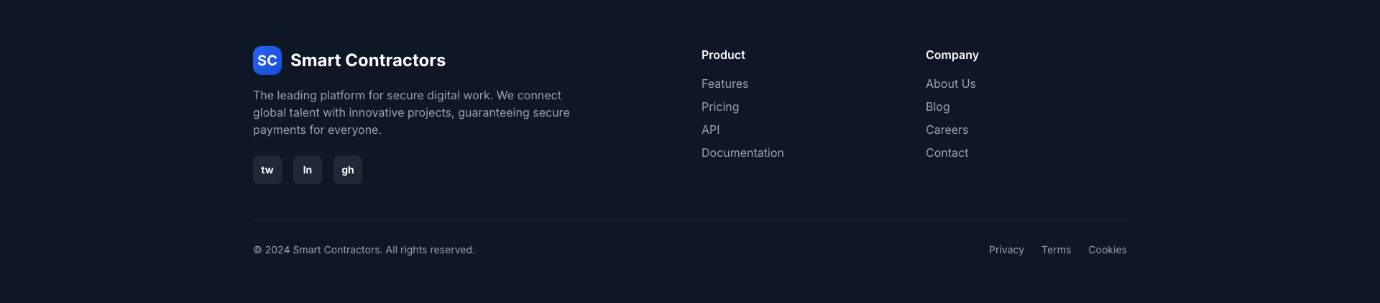




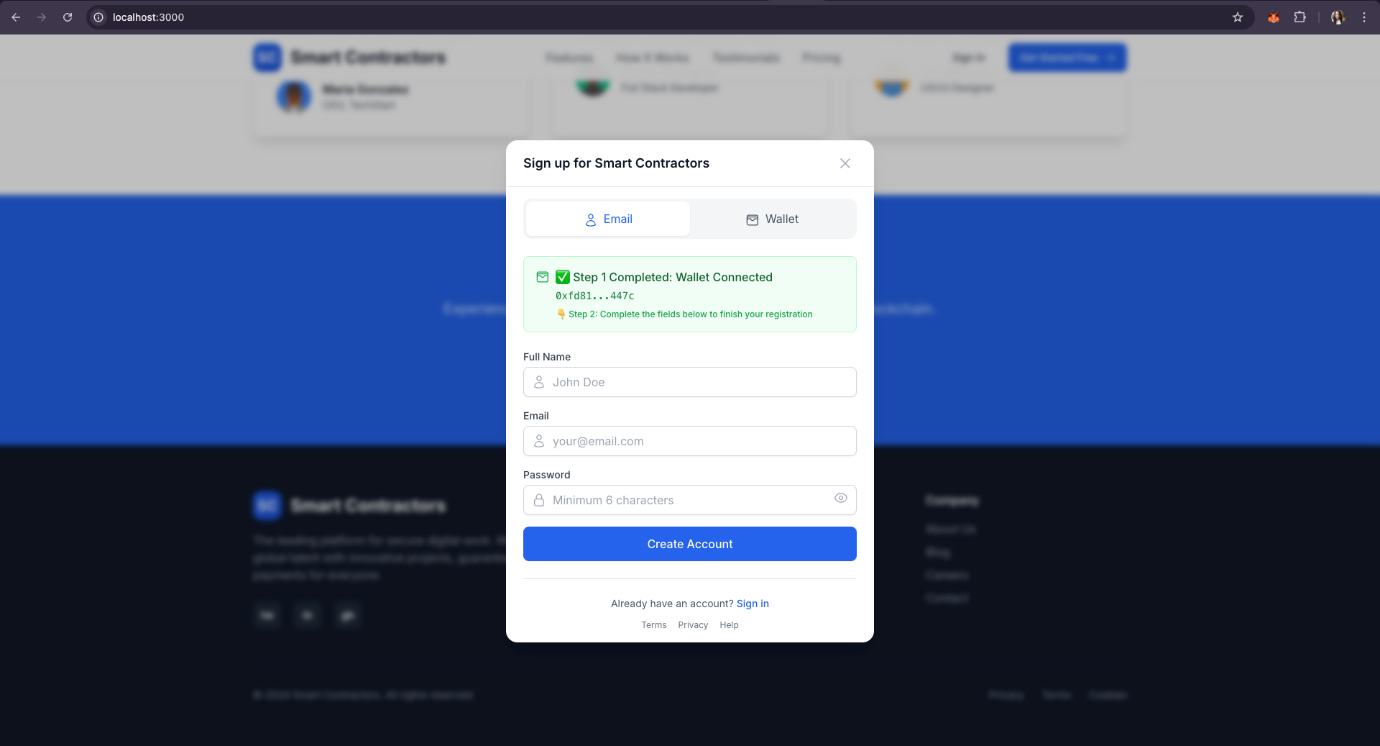




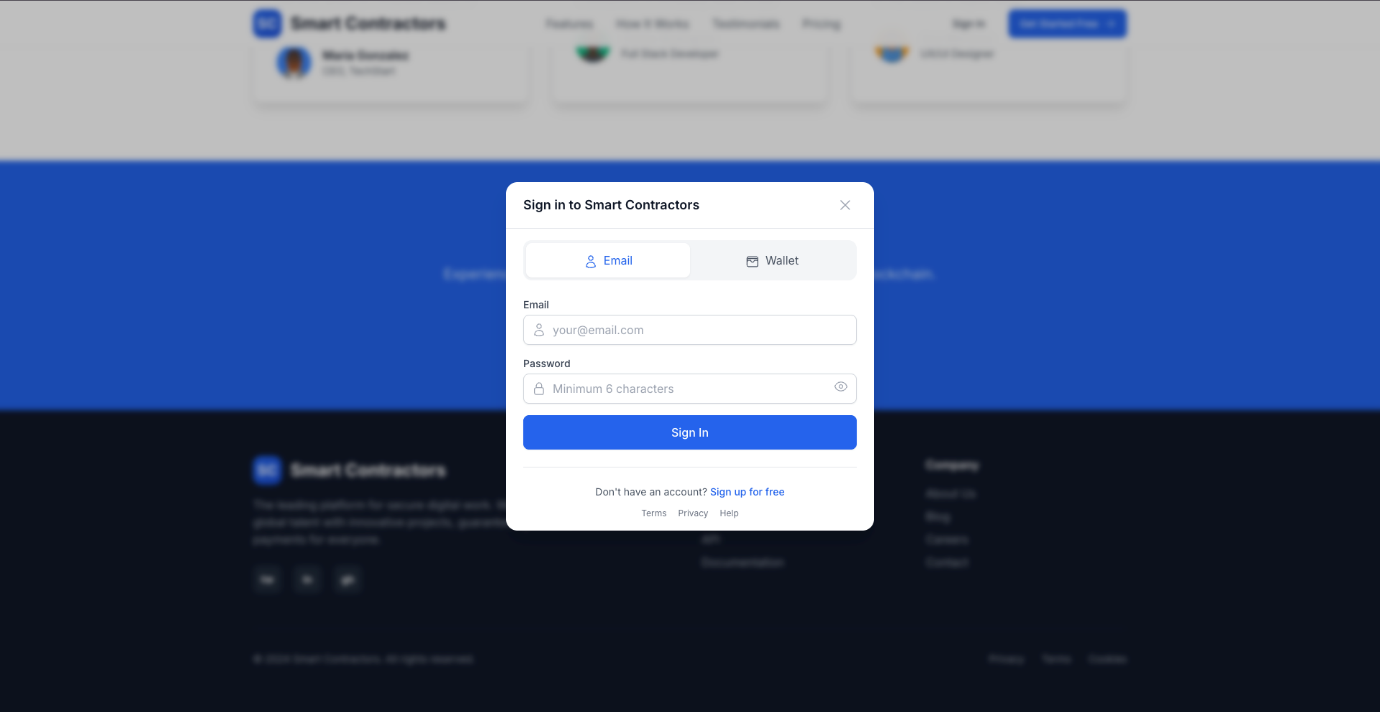


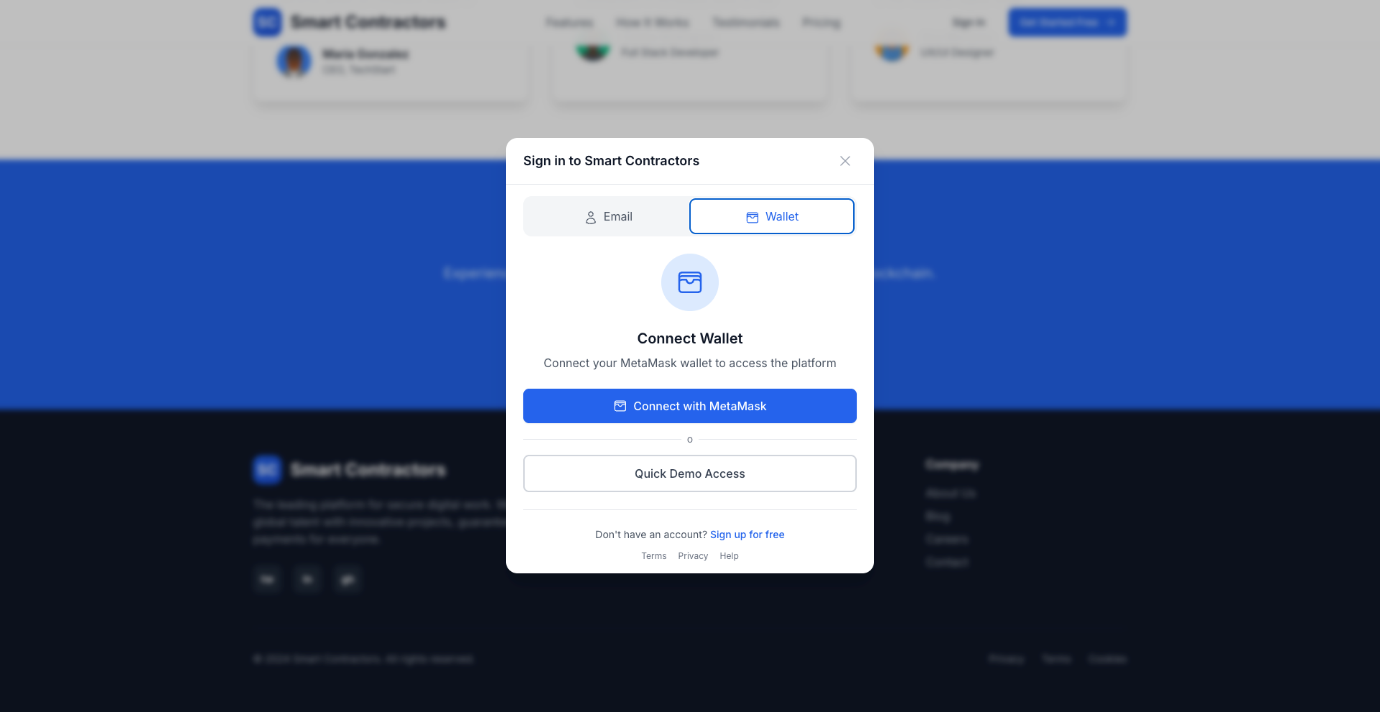


Registro



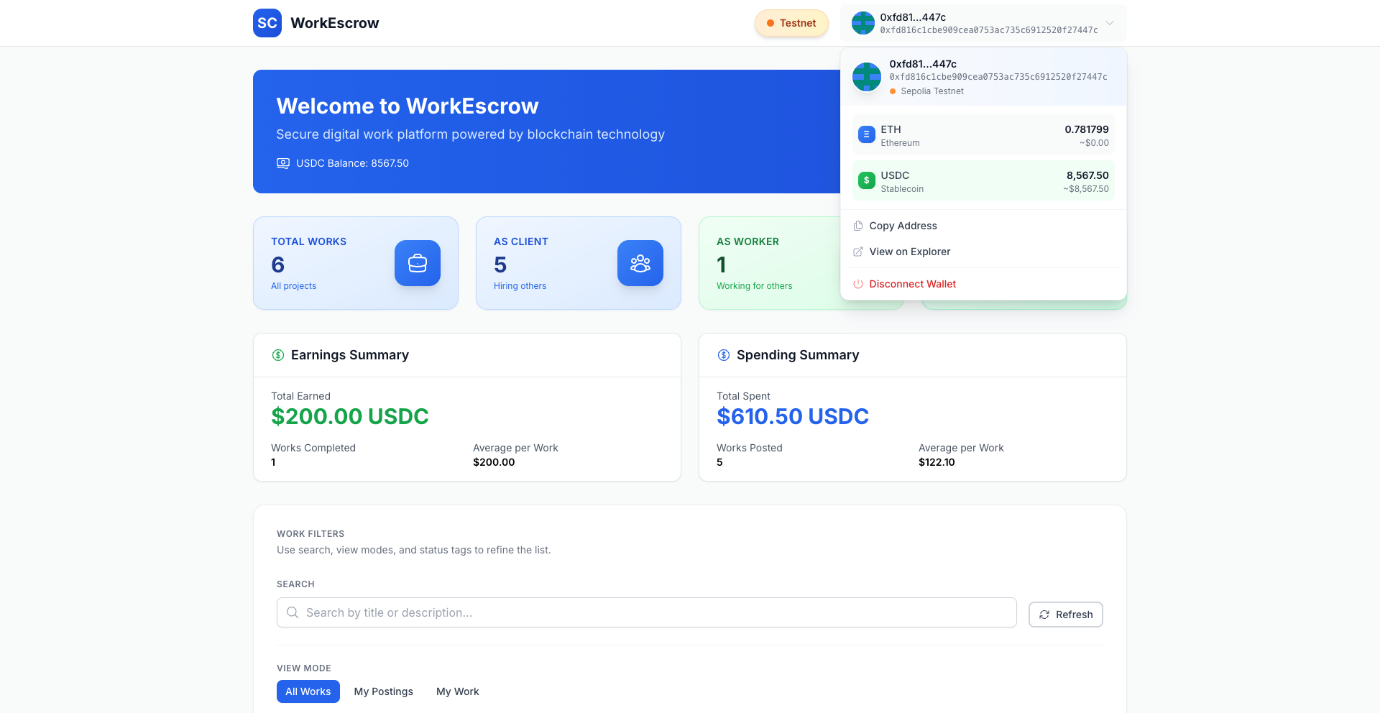
Login

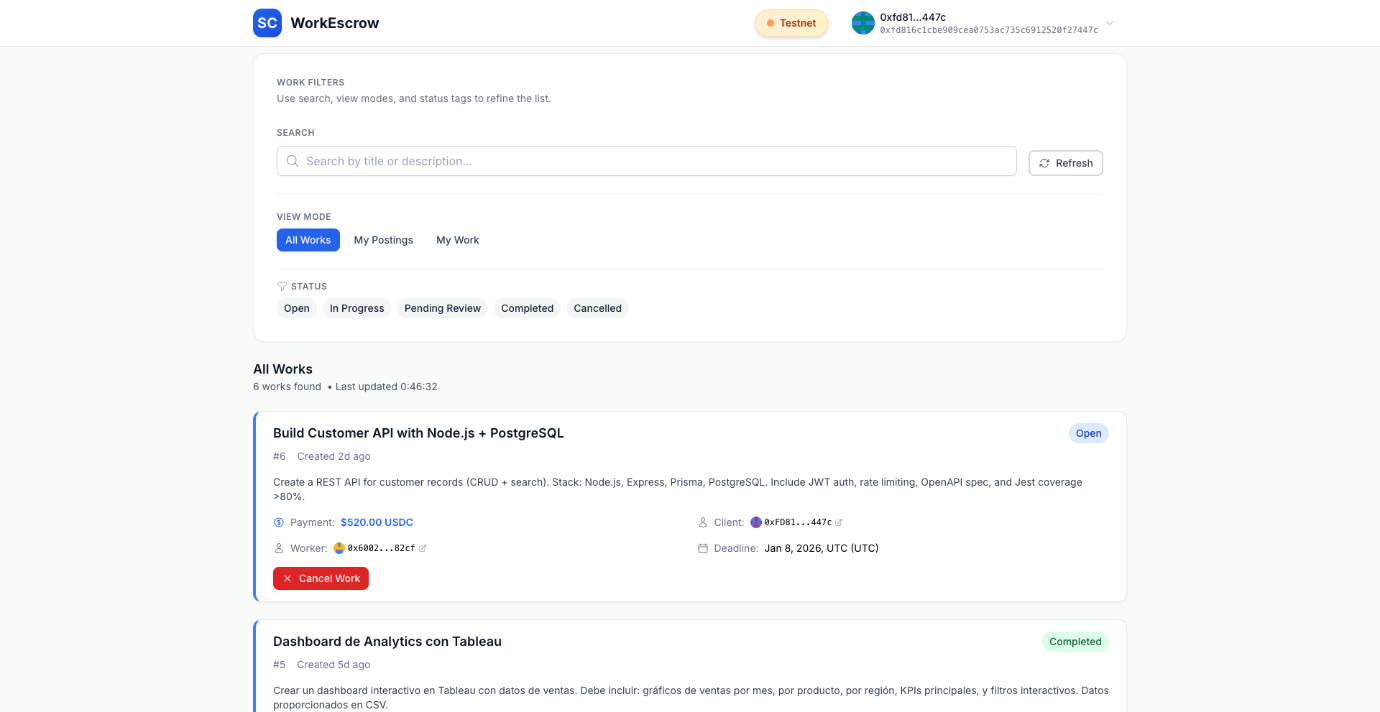


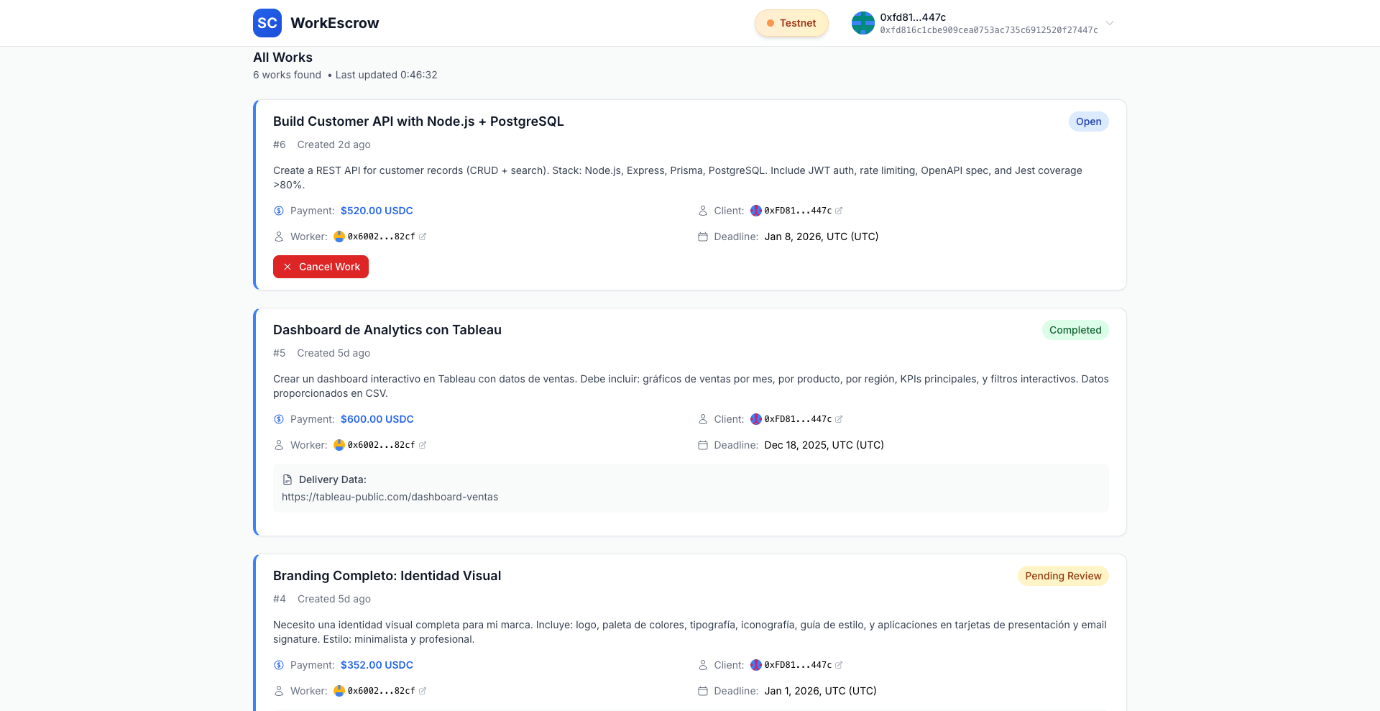


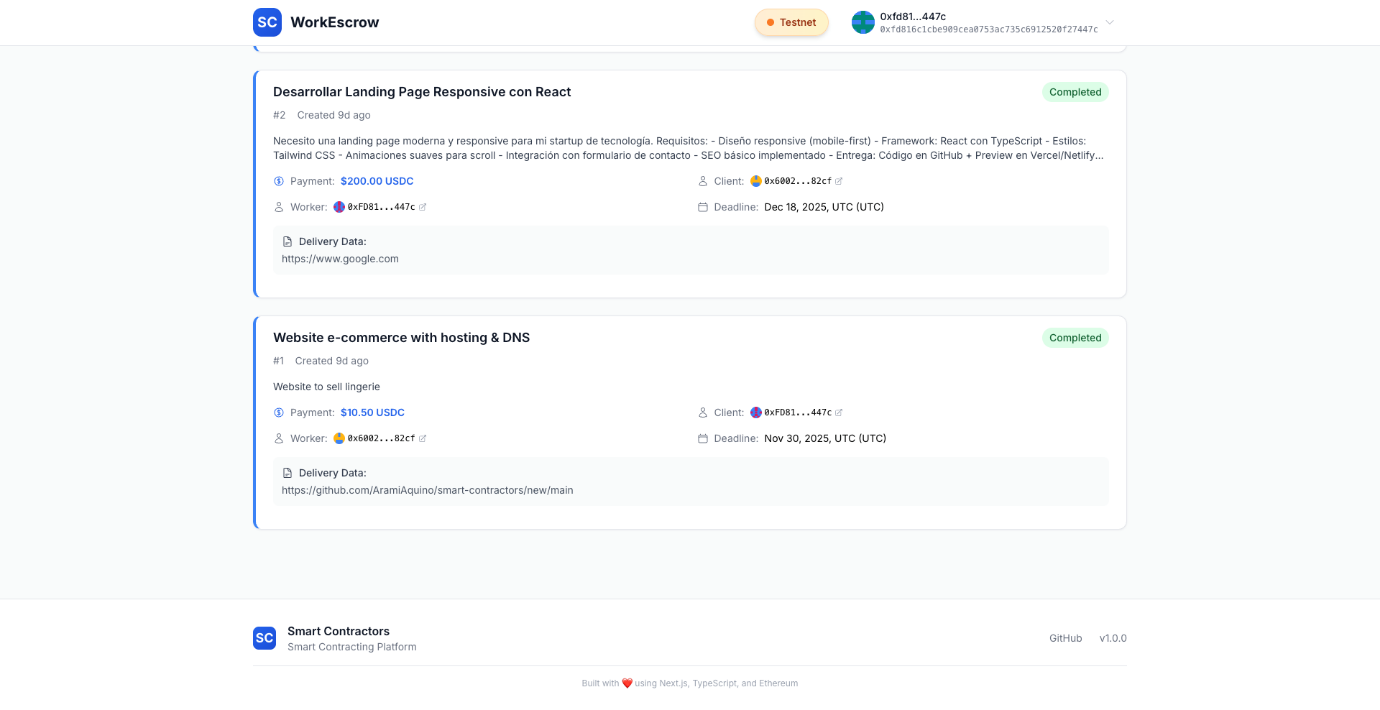
Home











# 6. Arquitectura de Software

## 6.1 Diagrama de Arquitectura

Capa de presentación (Frontend):

* Aplicación web en **React**
* Conexión a **MetaMask** (extensión de navegador)
* Manejo de formularios para contratos y entregas

*Capa Logica/Aplicacion (Backend)*

* API en **Node.js**
* Integración con blockchain usando **Web3.js**
* Gestión de repositorios (GitHub)

Capa de BlockChain:

* **Ethereum Testnet Sepolia**
* Contratos inteligentes en **Solidity**
* Patrón **Escrow** para manejar fondos en custodia
* Event logs para registrar cambios de estado

Capa de Infraestructura/Servicios:

* **Base de datos**
* **Circle USDC (Testnet)** como stablecoin de pago

## 6.2 Frameworks / Componentes utilizados

**Front-end:**

* React Next.js
* Web3.js
* MetaMask

**Back-end:**

* Node.js

**Blockchain:**

* Solidity (lenguaje de contratos inteligentes)
* Sepolia Testnet

**Base de Datos:**

* SQL

**Infraestructura/DevOps:**

* GitHub (repositorios + CI/CD)

## 6.3 Infraestructura tecnológica

|  |  |  |  |
| --- | --- | --- | --- |
| **Descripción** | *Plataforma descentralizada para la gestión de trabajo digitales con pagos seguros mediante contratos inteligentes en blockchain* | | |
| **Sistema Operativo** | *Windows, macOS* | | |
| **Bases de Datos** | *Blockchain de Ethereum* | *Sepolia Testnet* |  |
| **AppServer / WebServer** | *Node.js* | | |
| **Lenguajes utilizados** | *Solidity* | *TypeScript, JavaScript* | *HTML5, CSS3* |

*Se puede desplegar en cualquier infraestructura que soporte Node.js y Ethereum nodes. Para pruebas se usa Sepolia Tesnet con una Wallet en MetaMask*

## 6.4 Otros diagramas según corresponda

## 6.5 Estándar de codificación

|  |  |
| --- | --- |
| Categoria | Estandar / Buena practica |
| Solidity | Basado en [Style Guide — Solidity 0.8.31-develop documentation](https://docs.soliditylang.org/en/latest/style-guide.html) . Uso de pragma solidity ^0.8.x, converciones de nomenclatura (*PascalCase* para contratos, *camelCase* para funciones/variables) y comentarios *NatSpec* (///) para documentación interna |
| Seguridad y buenas practicas | **OpenZeppelin Contracts 5.4.0.** Que tiene: **ReentrancyGuard** (Proteccion contra ataques de reentrada), **Pausable** (Control de emergencia del contrato), **AccessControl** (Roles y permisos). Y cumplimientos del estandar **ERC20** para el manejo de tokens |
| Front (TypeScript + Node.js) | **ESLint + Prettier, TypeScript Style Guide (Airbnb).** Estructura **modular** con separacion de carpetas (contracts/, scripts/, test/) |
| Infraestructura Blockchain | Red **Ethereum** con **Sepolia Testnet** para validación en entorno controlado. Configuración compatible con **MetaMask** y **Hardhat Network** para despliegue local |
| Compatibilidad y escalabilidad | Arquitectura modular, adaptable a nuevas versiones de Solidity y librerías **OpenZeppelin**. Compatible con ecosistema **DeFi** y dApps del estándar **EVM** |
| Semantic Versioning | La semantica de la version es estructurada como **MAJOR. MINOR. PATCH** |

**Major**: Cambios grandes o incopatibles  
**Minor**: Nueva funcionalidad copatibles  
**Patch**: Correcciones o bugs

En nuestro caso, al ser la versión inicial estable, sería la **v1.0.0.** Teniendo la versión funcional y completa

# 7. Plan de pruebas

## 7.1 Diseño y ejecución de escenario de prueba

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Historia de usuario** | **Escenario de prueba** | **Resultado esperado** | **Resultado** | **Observaciones** |
| **US02 – Crear contrato inteligente** | Cliente con wallet conectada a Sepolia, saldo suficiente, ingresa monto, tiempo y descripción válidos, confirma en MetaMask | Contrato desplegado en blockchain, mensaje “Contrato creado exitosamente” | Aprobado/Fallo | Datos obligatorios Montos > 0, descripciones no vacías |
| **US02 – Crear contrato inteligente** | Cliente conectado, sin saldo suficiente, intenta confirmar | Sistema rechaza creación y muestra “Fondos insuficientes” | Aprobado/Fallo |  |
| **US02 – Crear contrato inteligente** | Cliente conectado en red incorrecta (ej. Goerli) | Sistema bloquea creación y muestra “Red no válida, conecte a Sepolia” | Aprobado/Fallo |  |
| **US08 – Aceptar contrato** | Freelancer con wallet conectada acepta contrato en estado “Creado” y confirma en MetaMask | Contrato cambia a “InProgress”, log en blockchain, mensaje “Contrato aceptado” | Aprobado/Fallo |  |
| **US08 – Aceptar contrato** | Freelancer intenta aceptar contrato en estado distinto de “Creado” | Acción bloqueada, mensaje “No está disponible para aceptar” | Aprobado/Fallo |  |
| **US08 – Aceptar contrato** | Freelancer intenta aceptar sin tener wallet conectada | Sistema rechaza y muestra “Debe conectar su wallet para continuar” | Aprobado/Fallo |  |
| **US10 – Enviar trabajo final** | Freelancer con wallet conectada, contrato en “InProgress”, carga archivo válido (PDF/JPG/PNG/ZIP ≤ 20MB), confirma transacción | Contrato pasa a “Submitted”, log en blockchain, mensaje “Entrega enviada” | Aprobado/Fallo | Validar límites y tipo de archivo |
| **US10 – Enviar trabajo final** | Freelancer intenta enviar entrega en contrato no “InProgress” | Sistema rechaza, mensaje “No se puede enviar entrega en este estado”. | Aprobado/Fallo |  |
| **US10 – Enviar trabajo final** | Freelancer carga archivo no permitido (ej. .exe) o vacío | Sistema rechaza carga, mensaje “Formato de archivo no válido” | Aprobado/Fallo |  |
| **US13 – Aprobar trabajo** | Cliente con wallet conectada aprueba entrega en contrato “Submitted” y confirma en MetaMask | Contrato pasa a “Completed”, fondos liberados al freelancer, mensaje “Entrega aprobada” | Aprobado/Fallo | Que el pago se refleje en wallet destino |
| **US13 – Aprobar trabajo** | Cliente intenta aprobar contrato en estado distinto de “Submitted” | Sistema bloquea acción y muestra “El contrato no está listo para aprobación” | Aprobado/Fallo |  |
| **US05 – Expiración automática** | Contrato con plazo de 7 días, freelancer no entrega | Al cumplirse el plazo, contrato pasa a “Expired”, fondos devueltos al cliente, log registrado, mensaje “Contrato expirado, fondos devueltos” | Aprobado/Fallo |  |
| **US05 – Expiración automática** | Contrato ya expirado | Sistema ejecuta devolución de fondos, log en blockchain. | Aprobado/Fallo |  |

## 7.2 Seguimiento de fallas

Para seguimiento issues, utilizar la herramienta que deseen y volcar la información aquí.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Descripción del Issue / Falla** | **Estado** | **Fecha de detección** | **Acción correctiva / Comentario** |
| F-01 | Bug en conexión de **WalletID**: no se actualiza correctamente el identificador al reconectarse la wallet. |  | 17/09/2025 | Se revisó la función de conexion de wallet, se agregará validación de reconexión y recarga de sesión. |
| F-02 | No muestra el estado al modificarse |  | 25/09/2025 | Posible error en listener de eventos StatedChange. Se revisará el socket del front. |
| F-03 | **Botón “Crear Work”** no responde cuando el formulario tiene campos vacíos. |  | 28/09/2025 | Se agregó validación previa al envío y mensaje de alerta en el front. |
| F-04 | El mensaje “Contrato expirado” no se muestra correctamente en algunos casos. |  | 30/09/2025 | Se revisa trigger del evento Se añadirá log y alerta visual. |
| F-05 | **Auth bug:** el backend no valida correctamente los tokens al iniciar sesión. |  | 27/09/2025 | Se agregará middleware de autenticación y validación de sesión en el backend. |
| F-06 | **Comunicación con blockchain:** error intermitente al interactuar con el contrato en la red de prueba. |  | 12/10/2025 | Se revisa la configuración del provider y los tiempos de respuesta de la testnet (Sepolia). Se implementará reintento automático. |

# 8. Manual Instalación

**Antes de empezar...**  
Para tener la información detallada de instalación, despliegue y testing, consultar la documentación técnica disponible acá:  
[Guía de instalación](https://github.com/Lenh22/smart-contractors/blob/main/DOCUMENTACION_COMPLETA.md#-para-desarrolladores-instalaci%C3%B3n-local)El sistema **WorkEscrow** utiliza **tecnología blockchain** sobre **Ethereum (Sepolia Testnet)** para gestionar trabajos digitales y pagos seguros mediante contratos inteligentes.  
El proyecto incluye un **backend en Solidity y Hardhat** y un **frontend en TypeScript/React** integrado con **MetaMask**

Requisitos:

* Node.js
* NPM
* MetaMask (extensión de navegador)
* Navegador compatible (Chrome o Brave)
* Cuenta de prueba en la red Sepolia (ETH de faucet)

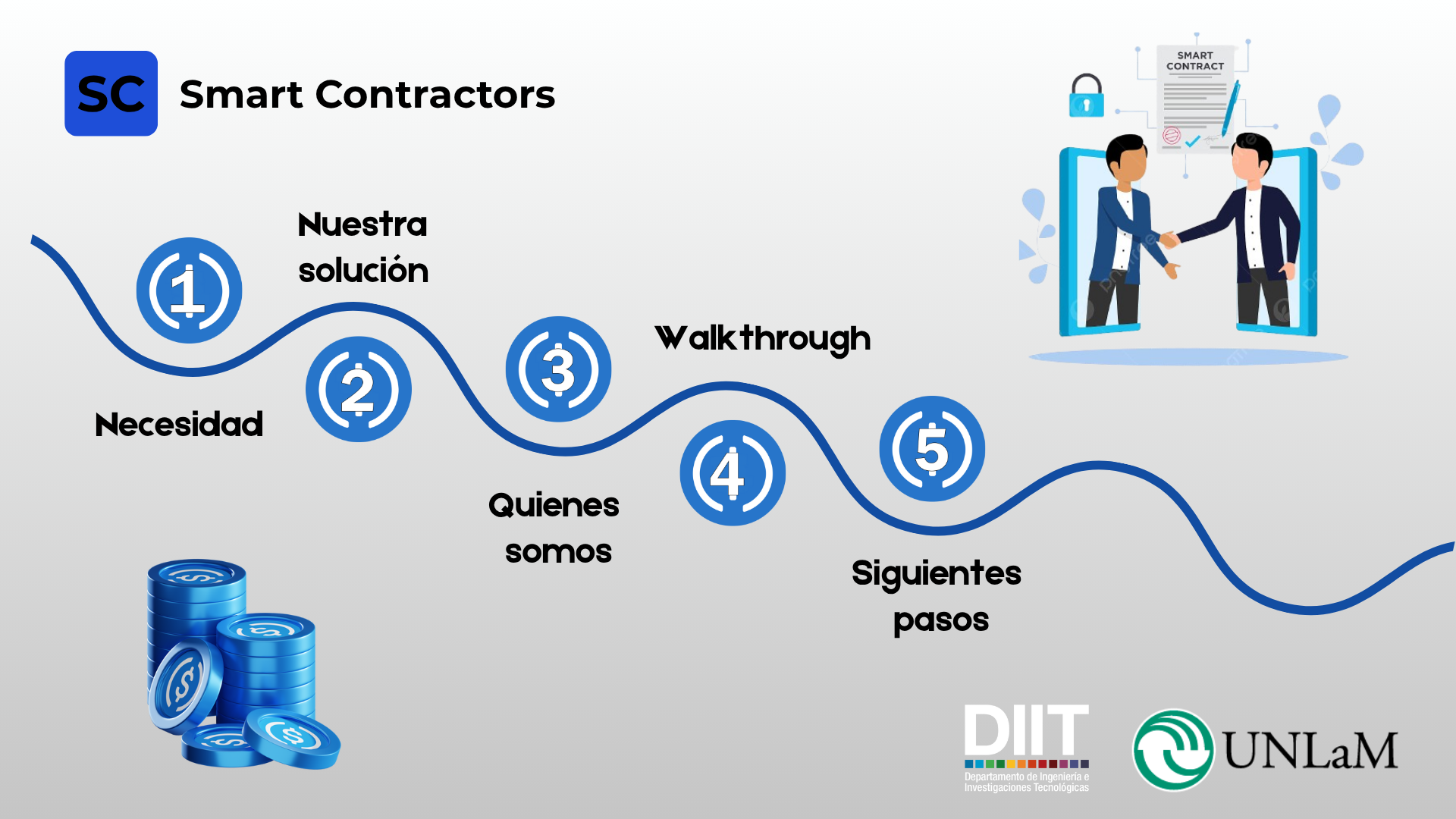
Instalación básica:

* Solicitar acceso al repositorio a [jaquinoromero@alumno.unlam.edu.ar](mailto:jaquinoromero@alumno.unlam.edu.ar)
* Clonar el repositorio desde Github
* Instalar dependencias con npm install.
* Compilar contratos con npm run compile.
* Ejecutar pruebas con npm run test.
* Desplegar en red local o en **Sepolia Testnet** con npm run deploy:sepolia.
* Ejecutar el frontend con npm run dev para abrir la app en <http://localhost:3000>.

Conexión de blockchain

* Conectar MetaMask a **Sepolia Testnet**.
* Configurar las variables de entorno (.env) con dirección del contrato y RPC.
* Asegurarse de contar con saldo de prueba (USDC/ETH) para operar.

# 9. Hoja de Ruta



# Anexos I. Retrospectiva

|  |  |
| --- | --- |
| 30-09-2025 - **Retrospectiva #1** | |
| **¿Qué hicimos bien?** | * Nos organizamos rápido como equipo y definimos roles * Se definió correctamente el flujo del cliente y los contratos inteligentes * Creamos diagramas |
| **¿Qué podemos mejorar?** | * Mayor comunicación entre back y front desde el principio * Endandarizar formatos de documentación inicial |
| **Ideas y propuestas** | * Usar Trello o Github para tareas * Crear una guía de instalación rápida compartida * Reuniones cortas de seguimiento |
| **ROI** (retorno de la inversión) *Indicar número del 2 al -2 de como fué la retrospectiva respecto al tiempo invertido* | 1  La inversión de tiempo fue buena, hubo algo de demoras por la curva de aprendizaje inicial |

|  |  |
| --- | --- |
| 14-10-2025- **Retrospectiva #2** | |
| **¿Qué hicimos bien?** | * Se logró integrar el frontend con MetaMask y la blockchain * Se logró el despliegue y testing de cada parte * El flujo del trabajador (aceptar trabajos, enviar entregas) quedó funcional * Se resolvieron errores del Sprint 1 (como walletID y comunicación backend) |
| **¿Qué podemos mejorar?** | * Mejorar la documentación del código y agregar comentarios * Falta optimizar la interfaz y mejorar la gestión de estados * Ajustar validaciones para evitar dobles clics o envíos duplicados |
| **Ideas y propuestas** | * Agregar validaciones en el contrato y una guía de instalación inicial * Centralizar el manejo de errores en el backend |
| **ROI** (retorno de la inversión) *Indicar número del 2 al -2 de como fué la retrospectiva respecto al tiempo invertido* | 2  Buen avance técnico y trabajo |

|  |  |
| --- | --- |
| 29-10-2025- **Retrospectiva #3** | |
| **¿Qué hicimos bien?** | * Se completó el flujo completo de cliente y trabajador, incluyendo aprobación y liberación de fondos * Se integraron correctamente la expiración automática * Mejoramos la coordinación del equipo y el seguimiento de bugs entre front y back * Los tests end-to-end confirmaron que el sistema funciona sin errores críticos |
| **¿Qué podemos mejorar?** | * Algunos plazos fueron ajustados sobre la marcha; mejorar la estimación de tiempos por tarea. * Consolidar la documentación técnica (README, guías de deploy y manual de instalación). * Faltó realizar más pruebas con diferentes usuarios en Sepolia. |
| **Ideas y propuestas** | * Implementar monitoreo automático para detectar fallas en los contratos. * Mejorar la UI/UX general del frontend y agregar feedback visual de transacciones. * Planificar una versión 4.0 con funcionalidades de disputa o IA evaluadora. |
| **ROI** (retorno de la inversión) *Indicar número del 2 al -2 de como fué la retrospectiva respecto al tiempo invertido* | 2  El equipo entregó la versión más completa, con un flujo funcional y estable que cumple los objetivos del proyecto |