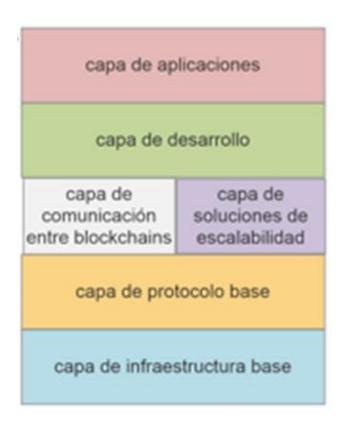
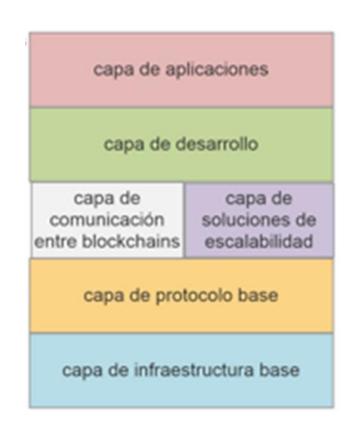
Capítulo 2

Decisiones de Diseño de Aplicaciones Distribuídas para Redes Blockchain

- ¿Para desarrollar una aplicación blockchain decidir en qué facilidades se apoya?
 - o Facilidades de la arquitectura de capas
 - Protocolos base, comunicación entre blockchains y soluciones de escalabilidad.
 - Infraestructura de desarrollo proporcionada por capa de desarrollo.
 - Otras aplicaciones
 - Almacenamiento descentralizado: IPFS, Filecoin
 - o Oráculos: ChainLink
 - Plataformas de pago: MoonPay



- Decisiones sobre las capas en que se apoya la aplicación distribuida:
 - Modelo de consenso que conviene usar
 - Elección de Plataforma de blockchain (protocolo base)
 - Estrategia de escalabilidad
 - Si se va a usar o no capa de comunicación entre blockchains.



Decisiones sobre el modelo de datos

- Decidir datos a almacenarse en la blockchain. Algunos ejemplos incluyen:
 - Transacciones: por ejemplo: transferencia de tokens entre usuarios, pagos realizados dentro de la aplicación, registros financieros transparentes.
 - Contratos inteligentes: código y lógica del contrato; resultados de la ejecución de los contratos.
 - Hashes de datos: en lugar de almacenar un archivo se almacena un hash criptográfico del mismo. Esto es para asegurar que el contenido no ha sido alterado.
 - Identidades y claves: identificadores únicos o claves públicas. Certificados digitales para autenticación.
 - Registros de eventos: información sobre eventos importantes de la aplicación, como logs de auditoría, cambios de estado o confirmaciones de acciones dentro del sistema.
 - Votaciones y decisiones: En aplicaciones de gobernanza se almacenan: votos emitidos por los participantes y resultados de las decisiones tomadas colectivamente.
 - **Propiedad y derechos**: registro de propiedad de activos digitales (p.ej. NFTs), derechos de uso o acceso asociados con los activos.

- Decisiones sobre el modelo de datos cont:
 - Decidir datos a almacenarse fuera de la blockchain.
 - Ejemplos de tipos de datos que suelen mantenerse fuera de la blockchain:
 - Datos voluminosos: como imágenes, videos, documentos o bases de datos extensas.
 Estos datos se mantienen en sistemas externos como:
 - Almacenamiento descentralizado: por ejemplo, IPFS o Storj, que son adecuados para compartir archivos de manera distribuida.
 - Almacenamiento centralizado: algunos proyectos recurren a servidores tradicionales para almacenar datos.
 - Datos confidenciales: información privada de los usuarios como direcciones, números de contacto, etc.
 - Procesos computacionales intensivos: aplicaciones que requieren cálculos complejos como aprendizaje automático o simulaciones suelen hacerlo fuera de la blockchain y solo registran los resultados relevantes en la cadena.
 - Datos temporales: como actualizaciones de precios, resultados deportivos, etc. se mantienen fuera.

Decisión sobre aplicaciones externas a usar

- Las dApps suelen apoyarse en herramientas y sistemas externos para complementar sus funcionalidades y superar algunas de las limitaciones inherentes a la blockchain.
- Principales categorías de aplicaciones externas:
 - Almacenamiento descentralizado: para almacenar grandes volúmenes de datos eficientemente mientras mantienen la seguridad y la descentralización: como IPFS (para archivos grandes y compartir contenidos de manera distribuida) y plataformas de almacenamiento en la nube descentralizadas (como Filecoin y Sorj).
 - Oráculos: para acceso a datos externos del mundo real. Por ejemplo, ChainLink.
 - Redes de computación: para realizar cálculos intensivos o procesar datos grandes. Por ejemplo: servicios centralizados en la nube (AWS Lambda o Google Cloud Functions), redes descentralizadas de computación colaborativa (p.ej. Golem).
 - APIs externas: por ejemplo: servicios de pago (PayPal o Stripe API), datos financieros en tiempo real (Yahoo Finance API, Alpha Vantage).
 - Gestión de identidad: para facilitar la autenticación de usuarios y manejo de identidades descentralizadas: Civic (soluciones de identidad y seguridad para usuarios), uPort (para manejar identidades digitales).
 - Almacenamiento centralizado: para almacenar archivos (Amazon S3, Google Cloud Storage)
 - Plataformas de análisis y visualización: para proporcionar insights a los usuarios. P.ej. Glassnode y Dune Analytics.

- Decisiones sobre la economía de tokens:
 - Economía de tokens: Muchas aplicaciones usan sus propios tokens.
 Interesa definir asuntos como:
 - Propósito: utilidad del token dentro de la app distribuida.
 - o Un token de utilidad está diseñado para acceder a ciertos servicios o funciones.
 - Un token de gobernanza permite a los poseedores participar en decisiones sobre el futuro de la aplicación, como votar en propuestas.
 - Un token de inversión: representa activos financieros, como acciones o derechos de propiedad.
 - Un token de recompensa: se usa para incentivar a los usuarios por acciones específicas.

Economía de tokens – cont:

- Suministro: se refiere a la cantidad total de tokens que existirán o que estarán disponibles en circulación.
 - Un token puede tener **suministro fijo**, otros pueden ser **inflacionarios** (se crean más tokens con el tiempo), o **deflacionarios** (se reducen los tokens disponibles)
- Distribución: se refiere a como se reparten los tokens entre los diversos participantes.
 - Distribución inicial: puede incluir eventos como una ICO (oferta inicial de moneda) o Airdrops (reparto gratuito).
 - Asignación: Una parte del suministro podría destinarse al equipo fundador, desarrolladores, inversores iniciales o comunidades.
 - Mecanismo de distribución: a través de minería, staking o recompensas por participar.

- Decisiones a tomar debido al uso de contratos inteligentes:
 - Propósito del contrato: objetivo del contrato; problema que resuelve; acciones que automatizará.
 - Funciones del contrato (transferir fondos, Validar datos, gestionar tokens, registrar transacciones, etc.) Definir parámetros de ellas.
 - Roles que interactúan con el contrato. ¿Cómo los distintos tipos de usuario interactúan con el contrato? ¿Quiénes acceden a qué funciones?
 - Datos del contrato: ¿qué almacenar?, estructuras de los datos, ¿qué datos almacenar en la blockchain y cuales no?
 - Reglas del contrato:
 - Las reglas definen el comportamiento del contrato.
 - Tipos de reglas:
 - o condiciones y restricciones,
 - o autorización: quien tiene permiso para ejecutar ciertas funciones del contrato,
 - Validaciones: controles para garantizar que las transacciones cumplen con las reglas establecidas.

- Decisiones a tomar debido al uso de contratos inteligentes cont:
 - Decidir acciones que se ejecutan automáticamente: los contratos inteligentes son autoejecutables, lo que significa que ciertas acciones se realizan automáticamente cuando se cumplen las condiciones. Por ejemplo:
 - o Transferencias automáticas (p.ej. liberar fondos si una tareas se completa)
 - Gestión de recompensas (calcular y asignar incentivos automáticamente)
 - Penalizaciones (deducir tokens o restringir accesos en caso de incumplimientos)
 - Mecanismos de seguridad: medidas para evitar vulnerabilidades y ataques.
 - Escalabilidad: el contrato puede manejar más usuarios o transacciones en el futuro?
 - Interacción con otras herramientas: oráculos para recibir datos externos, aplicaciones de almacenamiento descentralizados, etc.

- Decisiones a tomar debido al uso de contratos inteligentes cont:
 - Eventos generados por el contrato: los contratos pueden emitir eventos para informar a los usuarios o aplicaciones externas cuando ocurre algo relevante. Por ejemplo:
 - Registro de una transferencia de tokens
 - Notificaciones de actualización de estado en el contrato
 - Eventos útiles para aplicaciones descentralizadas que escuchan estos datos en tiempo real.
 - Interacción con otros contratos: en muchas aplicaciones distribuidas, los contratos inteligentes interactúan entre sí para lograr tareas más compleja. Por ejemplo:
 - Llamada a contratos externos: por ejemplo usar un contrato de oráculo para acceder a datos del mundo real.
 - Gestión modular: dividir la lógica en varios contratos para mejorar la organización y la escalabilidad.
 - Estándares como ERC-20 y ERC-721 siguen normas para garantizar la interoperabilidad entre contratos y aplicaciones.

• Decidir la estructuración de un contrato inteligente:

- Un contrato inteligente puede estructurarse de diferentes maneras según la complejidad, el propósito y los requisitos del proyecto.
- La organización del contrato tiene un impacto directo en su funcionalidad, seguridad y facilidad de mantenimiento.
- Algunas formas de organizar un contrato inteligente:
 - Contratos monolíticos: Toda la lógica del contrato se encuentra en un único contrato. Puede ser difícil de escalar o actualizar, puede ser más vulnerable a errores si el código es extenso.
 - Contratos modulares: se divide la funcionalidad en varios contratos que interactúan entre sí. Por ejemplo: un contrato principal para la lógica básica y contratos secundarios para funciones específicas (por ejemplo, manejo de usuarios y almacenamiento). Suele ser más sencillo de actualizar, mejora la organización del código. La interacción entre contratos puede aumentar el costo del gas.
 - Contratos heredados: usa la herencia para extender la funcionalidad de un contrato en base a otros contratos (p.ej. Solidity). Promueve reutilización de código, reduce errores.
 - Contratos basados en bibliotecas: se usan para funciones comunes.

- Decisiones para la gestión de seguridad e integridad
 - Modelo de gestion de identidad: como se manejan las identidades de los usuarios.
 - Identidad centralizada: usar servidores externos para autenticar usuarios.
 - Identidad descentralizada: implementar identidades auto-soberanas donde los usuarios controlan sus datos (p.ej. DID y verifiable Credentials).
 - Anonimato y pseudonimato: determinar si los usuarios pueden interactuar sin proporcionar datos personales.
 - Métodos de autenticación: como los usuarios acceden a la aplicación.
 Opciones:
 - Claves privadas: control de claves privadas para firmar transacciones.
 - Autenticación multifactorial: combiner claves privadas con otras medidas de seguridad como códigos temporales o biometría.
 - Tokens de acceso: usar tokens temporales para autorizar acciones dentro del Sistema.

- · Decisiones para la gestión de seguridad e integridad
 - Privacidad de los datos: garantizar que los datos de los usuarios estén protegidos. Opciones:
 - Cifrado: cifrar los datos sensibles que puedan ser manejados fuera de la blockchain.
 - Off-Chain storage: almacenar información privada (como identificadores o documentos) fuera de la blockchain y vincularla mediante hases.
 - Privacidad en transacciones: implementar tecnologías de privacidad para ocultar detalles transaccionales (p.ej. Tornado Cash).
 - Sistemas de recuperación: planificar cómo los usuarios recuperarán acceso en caso de pérdida de credenciales. Opciones:
 - Frases de recuperación: clave maestra para recuperar la cuenta.
 - Recuperación social: contactos de confianza ayudan al usuario a recuperar su cuenta.
 - Contratos multisig: exigir firmas multiples para recuperar credenciales.
 - Registros de actividad y auditoría: decidir qué datos se registrarán para monitoreo. Por ejemplo, usar logs de actividad (rastreo de acciones de usuarios y contratos para detectar anomalías.)